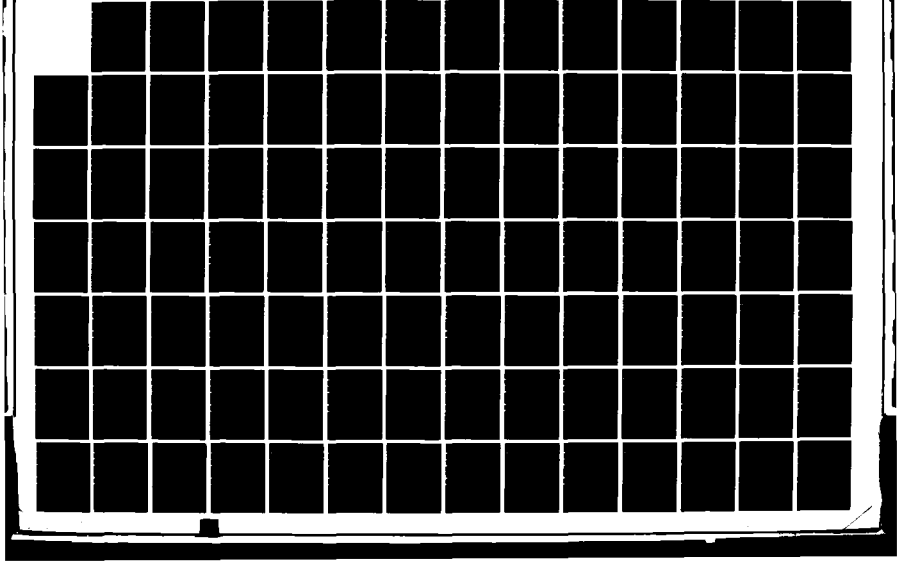
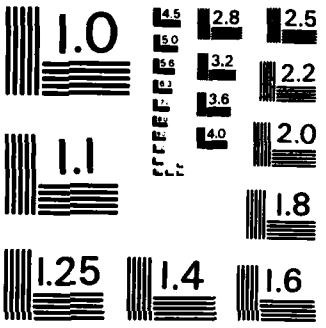
AD-A145 807 RESEARCH IN KNOWLEDGE REPRESENTATION FOR NATURAL LANGUAGE UNDERSTANDING(U) BOLT BERANEK AND NEWMAN INC CAMBRIDGE MA C SIDNER ET AL. SEP 84 BBN-5694 1/3

UNCLASSIFIED N00014-77-C-0378 F/G 5/7 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS - 1963 - A

Bolt Beranek and Newman Inc.

Report No. 5694

Research in Knowledge Representation For Natural Language Understanding

Annual Report
1 September 1983 to 31 August 1984

Prepared for:
Defense Advanced Research Projects Agency

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER BBN Report No. 5694	2. GOVT ACCESSION NO. AD-A145 807	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) RESEARCH IN KNOWLEDGE REPRESENTATION FOR NATURAL LANGUAGE UNDERSTANDING		5. TYPE OF REPORT & PERIOD COVERED Annual Report 9/1/83 - 8/31/84
		6. PERFORMING ORG. REPORT NUMBER BBN Report No. 5694
7. AUTHOR(s) Sidner, C., Goodman, B., Haas, A., Moser, M., Stallard, D., Vilain, M.		8. CONTRACT OR GRANT NUMBER(s) N00014-77-C-0378
9. PERFORMING ORGANIZATION NAME AND ADDRESS Bolt Beranek and Newman Inc. 10 Moulton Street Cambridge, MA 02238		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Office of Naval Research Department of the Navy Arlington, VA 22217		12. REPORT DATE September 1984 13. NUMBER OF PAGES 186
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Distribution of this document is unlimited. It may be released to the Clearinghouse, Department of Commerce, for sale to the general public.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Artificial intelligence, natural language understanding, knowledge representation, semantics, semantic networks, KL-TWO, NIKL, belief and knowledge, reasoning, RUP, syntax, parsing, reference, miscommunication, plan recognition, speaker intention, planning, situation calculus, discourse, semantic acquisition, HSP, IRACQ

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

BBN's ARPA project in Knowledge Representation for Natural Language Understanding is aimed at developing techniques for rendering computer-based assistance to a decision maker who is attempting to understand and react to a complex, evolving situation. This report summarizes our research in knowledge representation and natural language over the past three years. In particular, we discuss the development of KL-ONE, NIKL, and KL-TWO, the advances in RUS and the PSI-KLONE systems, and our work in understanding discourse and speaker's intentions. The report also contains a

cont'd.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Abstract (cont'd.)

number of papers on different areas of our research: the interface of KL-TWO, the NIKL environment, semantic acquisition, a model of planning with time and events, recognizing intentions in discourse and understanding references in the face of miscommunication. We also document publications and presentations by members of the research group over the past year.

Report No. 5694

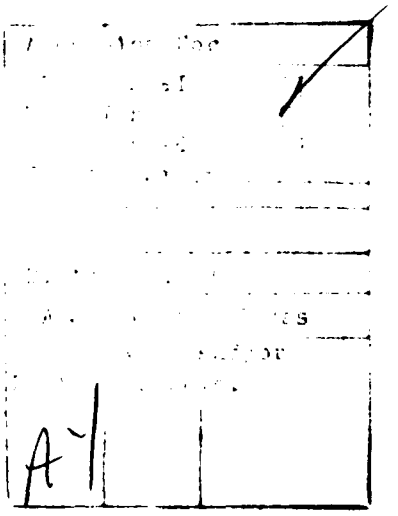
RESEARCH IN KNOWLEDGE REPRESENTATION FOR NATURAL LANGUAGE UNDERSTANDING

Annual Report
1 September 1983 - 31 August 1984

September 1984

Principal Investigator
Candace L. Sidner

Co-principal Investigator
Robert J. Bobrow



Prepared for.

Defense Advanced Research Projects Agency
1400 Wilson Boulevard
Arlington, VA 22209

ARPA Order No. 3414

Contract No. N00014-77-C-0378

Effective Date of Contract
1 September 1977

Contract Expiration Date
30 September 1984

This research was supported by the Advanced Research Projects Agency of the Department of Defense and was monitored by ONR under Contract No. N00014-77-C-0378. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

BBN's ARPA project in Knowledge Representation for Natural Language Understanding is aimed at developing techniques for rendering computer-based assistance to a decision maker who is attempting to understand and react to a complex, evolving system or situation. The decision maker's access to the situation is mediated by an intelligent graphics display system, which is controlled largely through natural language input. A typical and motivating instance is that of a military commander in a command and control context, either of strategic situation assessment or of tactical crisis management. In such situations, the commander requires a flexible and easily controllable system capable of manipulating large amount of data and, most importantly, of presenting information in a variety of forms suited to the user's expressed or inferable needs and capacities.

A display system of the kind envisaged would have the capacity to present information in tabular, graphical, textual, and perhaps cartographic forms. The user of such a system must be able to monitor, add, change and delete information and, independently, to create and alter the various representational forms. Moreover, for the system to be truly flexible and adaptive, it must maintain models of the domain (situation) being represented, of the representational systems at its disposal, and of the user's conceptions of these domains, situations, and systems of representation. For this last purpose, the system must also be able to construct models of its interactions with the user.

On the basis of these different kinds and sources of information, the system

must produce intelligible and appropriate displays in response to high-level descriptions and commands. That is, the commander can usually be expected to request a presentation of certain aspects of the situation or system being monitored in terms appropriate to the domain itself and not in terms of display forms. Even when the request, explicit or implicit, is expressed in terms of display forms, the specification will typically be at a level of abstraction appropriate to the commander's purpose - not to those of a graphics system designer or programmer. The system must be able to accept a description of the information to be represented together with an abstract specification of a display-type and then it must intelligently determine the details required actually to produce an effective display. Finally, given information about the user's knowledge of the situation being monitored and his particular concerns with respect to it, the system must, in some cases, be able to infer what kind of display a user might want to see, produce it and monitor the user's response to its initiatives

The crucial requirements for a medium of communication with such a system are robustness, flexibility, the ability to express specifications while abstracting from details of various kinds, and the ability to express conceptualizations of both the presented domain and the modes of display in ways that match a user's conceptualization. By far the most natural form of access to and control over such a system for most users will be through the use of natural language input. Hence, a major focus of our research has been the design of a system powerful enough to represent the content of natural language utterances together with facts about the user's beliefs and goals as these are communicated in the user's interactions with the system. Such a representational formalism must also express, in usable form,

information about the domain or situation being monitored and the nature of the display system itself.

Knowledge Representation

The development of knowledge representation tools is a major accomplishment of our research group. We have extended and revised the KL-ONE knowledge representation system to a new system KL-TWO [1] It incorporates

1. A taxonomic knowledge representation system of the KL-ONE variety called NIKL (for a new implementation of KL-ONE), a cleaner and more efficient version of our earlier KL-ONE (see BBN Report No. 5421).
2. A propositional reasoning system called PENNI based on RUP, a TMS originally designed by McAllester [1] and reimplemented in INTERLISP (see BBN Rep No. 5421)
3. An interface between these two systems.

KL-TWO represents our first implementation of a hybrid system, that is, a system built from several components, each of which provides an expressive, natural representation, and a special-purpose reasoning engine for efficient control of inference. each reasoner is then interfaced to the others. Hybrid systems include constructs that allow a user to capture important intuitions about the structure of a domain and also offer the user more control over the reasoning process than previously possible. In KL-TWO, we have joined a reasoner for taxonomic hierarchies of knowledge with one for propositional logic Our recent experience in designing and implementing an interface between these two reasoners is reported in

---

[1] In a previous report we used the term NIKL to refer to the new version of KL-ONE and to the whole system that interfaces assertions with the new KL-ONE implementation. KL-TWO is the current name for the whole system.

Our experience with NIKL over the past year has led us to develop a programming environment for working with NIKL taxonomies. This environment makes it possible to build, edit, and install new and changed hierarchies and makes use of several Interlisp debugging features for NIKL. The programming environment, developed by D. Stallard, is reported on in section .

Explorations in Planning

Planning has become a focus of our research efforts largely through our natural language research on discourse (see the discussion below) where it is necessary to come to an understanding of the speaker's plans and to develop a plan for responding to the speaker s intentions. We have recently begun to formalize some research on planning for use in our group. The paper in section by A. Haas presents a theory of planning involving time and agents that we believe will solve some of the problems that have limited the plans a planner can design.

Natural Language Understanding

Our natural language research has made several significant advances. We have extended the RUS and PSI-KLONE system to include tools to aid users in adding lexical information without knowledge of the implementation (see BBN Report No. 5421) and in adding semantic rules (see Section in this report). An important aspect of lexical acquisition is the changes to the lexicon we have made for subcategorization of verbs. BBN Report No. 5684 presents an explanation of subcategorization of verbs that motivated the restructuring of the dictionary. Finally, we have continued to develop the RUS parsing system (see BBN Report No. 5188) and to make a version of RUS that is transportable for a variety of decision-support systems (see BBN Report No. 5421).

Since language understanding also requires understanding what the meaning of a utterance tells the hearer about the world. we have explored several issues in discourse understanding. Building on earlier work in recognition of speaker's intentions (see Chapter 5.1 in [2]. we have implemented a plan parser for recognizing speaker s plans and explored its relation to other discourse behavior. This work is reported in this volume (in6). We have explored the general requirements on building a natural language system with graphics (see Report No 5242).

Understanding descriptions plays a significant part in discourse and natural language as a whole. and we have investigated it in two ways. a general framework for understanding the first use of descriptions in discourse (see Report No. 5421. Sec. 9). and a theory. and implementation of (extensional) referential descriptions to objects in the hearer s view (See Report No 5421. Section 10. and in this volume). We hope to extend the general framework for descriptions in future research.

The theory and implementation of extensional reference encompasses a model of reference that differs significantly from previous approaches. In this theory. a reference process doesn't simply succeed or fail. Rather the process takes into account ways the speaker may have miscommunicated about the object of interest. and it relaxes aspects of the description until a referent is found. By viewing communication between speaker and hearer as a process that may include description errors. Goodman has taxonomized many kinds of miscommunication in discourse other than those involving reference (see the forthcoming BBN Report No. 5681). We expect to expand our research in miscommunication and discourse to include a model of some of these discourse errors.

Finally research by Haas (see Section 6 in BBN Report No 5421) has provided us with some additional tools for reasoning about belief While belief is a part of knowledge representation research in general, it is particularly valuable for our discourse research because we must model the beliefs that speakers and hearers have about each other

The last section of this report contains a list of papers published this year, papers to appear and presentations during the contract year October 1, 1983 to September 30, 1984

REFERENCES

[1] McAllester, D. A.
An Outlook on Truth Maintenance
AI Memo 551, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, August, 1980.

[2] Sidner, C.L., Bates, M., Bobrow, R.J., Brachman, R.J., Cohen, P.R., Israel, D., Schmolze, J., Webber, B.L. and Woods, W.A.
Research in Knowledge Representation for Natural Language Understanding. Annual Report 1 September 1980 – 3 August 1981
BBN Report No 4785, Bolt Beranek and Newman Inc., Cambridge, MA, 1981.

2. KL-TWO, A HYBIRD KNOWLEDGE REPRESENTATION SYSTEM

Marc Vilain

2.1 Introduction

This paper is concerned with how separate knowledge representation systems can be combined so as to provide a more usable system. The paper focuses on one such system, the KL-TWO system. KL-TWO contains two components that complement each other: one, called NIKL is used to define the meanings of predicates, and the other, called PENNI, is used to assert propositions about the world. This paper outlines features of NIKL and PENNI, and shows how they are interfaced.

2.2 Hybrid Reasoners

Knowledge representation systems are required to provide seemingly contradictory features. A good knowledge representation system should provide representational generality. I should be able to use it to represent any information my programs need. The system should have a well-defined semantics, so that I can be sure that my statements capture the meanings I intended. Finally, the system should be computationally efficient: it has to be relatively fast so that my programs that use it can accomplish honest work.

There are several well-established approaches to knowledge representation that attempt to provide some of these features. Full theorem provers for first-order languages certainly provide generality and precise semantics, but at the expense of computational efficiency. Frame-based systems are more efficient, but they give up

precise semantics and generality There are other examples, but generally knowledge representation systems give up one or another of the features I listed above in favor of one or more of the others Is it really true, then, that these features are incompatible?

A recent trend to reconcile these features is to build knowledge representation systems as hybrid reasoners In this approach, the system is composed of several cooperating reasoners Each of the reasoners is intended to work on some part of the knowledge representation task, each provides an inference procedure for some restricted sublanguage The language of the system as a whole is thus the union of the sublanguages Given sublanguages with well-defined semantics and efficient reasoners for these sublanguages, the designer of a hybrid system must interface these subcomponents so that their interaction fully exploits the capacities of each individual reasoner If this can be done, then the system as a whole can be expected to provide the efficiency and precise semantics inherent in its subcomponents, as well as a greater generality than each component could provide by itself

Several recent knowledge representation efforts have followed this approach, or one very similar, among them are CAKE [13] and KRYPTON [5] This paper describes another hybrid system which my colleagues and I have been developing recently Our system is called KL-TWO

2.3 KL-TWO at a Glance

KL-TWO is so named in deference to the earlier knowledge representation system KL-ONE [1, 2, 14, 3]. Our work on KL-TWO is a natural outgrowth of the earlier research. the new system embodies many of the ideas that motivated KL-ONE. The principal notion that was inherited in this way is the distinction drawn by KL-ONE between terminological reasoning and assertional reasoning [14, 4].

Terminological reasoning is reasoning about the *terms* (or more precisely, the predicates) used to describe the world The terminological component of KL-ONE provided ways to define and interrelate predicates. For example, in KL-ONE I could encode such things as:

"a PARENT is a PERSON with one or more OFFSPRING"
"a FATHER is something which is both a PARENT and a MAN"

These are assertions about the nature of the predicates PARENT and FATHER. their meaning is being defined in terms of the predicates PERSON, OFFSPRING, and MAN. The first assertion, for example, makes the PARENT predicate into a *structured predicate*, a complex predicate that corresponds to the expression "PERSON with one or more OFFSPRING" One way to view terminological reasoning is as a process of making the inferences that follow from terminological assertions. For instance, it follows from my two assertions above, that the meaning of the FATHER predicate encompasses the meaning of the PERSON predicate In other words, anything which can be described as a FATHER can be described as a PERSON

The domain of terminological reasoning is the set of structured predicates used to describe the world. In terminological reasoning, no assertion is made about the

actual existence of individuals that satisfy these predicates. such propositions are the domain of assertional reasoning It is in the assertional component of KL-ONE that I would have stated propositions about the world. For example.

"John is a MAN"
"John has an OFFSPRING Mary"

The propositions in the assertional component can be seen as applying the structured predicates of the terminological component to individuals in the world

KL-TWO maintains this terminological/assertional distinction. Like its predecessor, KL-TWO has two components. one for reasoning about *S*tructured predicates (the *S*-component). and one for reasoning about *P*ropositions (the *P*-component). The first is embodied in a program called NIKL. and the second in a program called PENNI. KL-TWO's hybrid nature arises from the fact that NIKL and PENNI are entirely distinct subsystems. each with its own sublanguage and deduction algorithm. The two subsystems cooperate to complement each other s capacities.[2]

The rest of this document describes KL-TWO's subsystems. I will cover some of the features of the P-component and of the S-component. and show how the two are interfaced.

---

[2]Note that these are only the first two components of the system that have been interfaced. We will soon add a temporal reasoner similar to the one in [18]. Other components are planned as well.

2.4 The P-component

PENNI, the P-component of KL-TWO is a modified version of a program written by David McAllester of MIT. His program is the Reasoning Utility Package -- RUP for short [10, 11]. The features of the P-component I will describe in this section are originally features of RUP.

At the heart of the P-component is a database of propositions. The language of this database is the quantifier-free predicate calculus with equality. This language is defined as containing:

1. Expressions of the form (P x), (Q x y).... as in (MAN John) or (OFFSPRING John Mary).
2. Expressions of the form (= x y), as in (= (grade Bill) B+).
3. Boolean combinations of (1) and (2), as in (=> (FATHER John) (PERSON John)).

This language doesn't contain any quantifiers. I will say more about this below.

The database permits incremental assertions and retractions, and is built as a truth maintenance system (or TMS). The P-component's database qualifies as a TMS in part because it always records how it arrives at a conclusion. More precisely, when an assertion is added to the database, it causes a limited number of deductions to be made in a forward-chaining way. As each deduction is made, the inference algorithm maintains an explicit record of the antecedents that were used in the deduction step. For example, say the database contains the proposition (=> P Q). Adding the proposition P causes Q to be deduced. further, the consequent of this inference, namely Q, is explicitly supported by pointers to its antecedents, P and (=> P Q).

These explicit records serve to provide the P-component with several important features that are characteristic of truth maintenance systems. These include the ability to justify a deduction by returning the exact set of user-asserted propositions that entail it, the ability to retract efficiently the logical consequences of a proposition when that proposition is itself retracted, and the ability to perform fast dependency-directed backtracking. These features are extraordinarily useful; eloquent arguments for them have been made by McAllester [10, 11], Doyle [7], and others.

I mentioned above that the language of the P-component does not include quantification. The decision not to include quantifiers was made by McAllester when he designed RUP, and follows from his view of logical reasoning as having two aspects: deduction and instantiation. Deduction is the process of deriving the consequences of a set of propositions, whereas instantiation is the process of applying quantified sentences to produce new propositions. The approach embodied by *RUP* (and *hence by* our P-component) is to focus on providing a good efficient mechanism for deduction,[3] and to remain uncommitted as to how to perform instantiation.

The choice of how to instantiate quantified sentences is left up to the user. This is facilitated by providing "hooks" into the database that allow the user to augment the language of the P-component. The hooks are realized by the mechanism

---

[3] RUP's forward-chaining deduction algorithm is very efficient. It is incomplete, however: it fails to perform certain inferences, roughly those that require case analysis. RUP does provide another deduction algorithm which extends the first one to produce a complete proof procedure.

of demon invocation. RUP provides several varieties of IF-ADDED and IF-NEEDED demons The version of RUP used in KL-TWO contains McAllester's demon invocation mechanisms. However, much of what is interesting about KL-TWO is that the system can provide a class of quantified inferences efficiently and automatically, without requiring the use of demons. These inferences are performed by KL-TWO's second subreasoner, the S-component of the system.

2.5 The S-Component

I mentioned earlier that NIKL, the S-component of KL-TWO, has as its domain the predicates used to form propositions in the P-component. The language of the S-component allows one to state how these predicates are interrelated, and allows one to define new structured predicates in terms of other predicates. In this section I will only discuss a subset of the S-component language, the full language is covered in [12, 15].

There are two types of S-component expressions of interest to this paper, 1-place predicates and 2-place predicates. These have traditionally been referred to as *Concepts* and *Roles* respectively. I will use these two terms frequently. The S-component language provides constructors that combine predicates to produce new structured predicates I will describe three of these constructors. CMeet, CMin, and CRestriction.[4] All three combine Concepts or Roles to produce new Concepts.

---

[4]Out of context this choice of constructors may seem arbitrary. [1, 15, 8] show how they can be used to capture many of the intuitions of semantic nets.

The simplest of these constructors is CMeet. CMeet is used to combine several 1-place predicates (say n of them) to produce the 1-place predicate that corresponds to the conjunction of the original n predicates. CMeet has the following definition.

(CMeet C1 C2 ... Cn) $\equiv \lambda$x. (C1 x) & (C2 x) & ... (Cn x)

For example, I can use CMeet to define the predicate FATHER as the predicate which is true of those things for which both the PARENT and MAN predicates are true.

FATHER $\equiv$ (CMeet PARENT MAN)
$\equiv \lambda$ x. (PARENT x) & (MAN x)

This can be glossed as "a FATHER is something which is both a PARENT and a MAN".

Viewing lambda-expressions as predicates may seem unintuitive at first. These lambda-expressions are to be understood as complex structured predicates that are transformable into (open) sentences by predicate application.[5] This allows giving definitions such as the preceding one a more familiar reading. In this case

$\forall$x (FATHER x) <=> ((PARENT x) & (MAN x))

The two other constructors I mentioned, CMin and CRestriction, combine Roles and Concepts to produce new Concepts.[6]

(CMin R n) $\equiv \lambda$ x. $\exists$ n y (R x y)
(CRestriction R C) $\equiv \lambda$x. $\forall$y ((R x y) => (C y))

---

[5] See [15] for a more detailed exposition of this point.

[6] In these definitions R is a Role, C is a Concept and n is an integer. The expression n y (R x y) is a numerical quantifier: the expression is an abbreviation for $y_1 \ldots y_n$ (R x y_1) & ... & (R x y_n), where the y_i are distinct.

The expression (CMin *R n*) denotes the Concept (i.e., the 1-place predicate) which holds true of those things which are *R*-related to at least *n* other things. The Concept corresponding to (CRestriction *R C*) is true of those things which are only *R*-related to things which are "*C*-ish" (i.e., for which the *C* predicate holds).

These constructors are best illustrated with some examples. I can define PARENT in terms of PERSON (a Concept) and OFFSPRING (a Role) with CMeet and CMin.

PARENT $\equiv$ (CMeet PERSON (CMin OFFSPRING 1))
$\equiv \lambda x.$ (PERSON x) & $\exists y$ (OFFSPRING x y)

Again, there is a more familiar reading for this definition

$\forall x$ (PARENT x) <=> (PERSON x) & $\exists y$ (OFFSPRING x y)

This can be glossed as "a PARENT is a PERSON with at least one OFFSPRING".

I can define MAN in terms of PERSON (a Concept), GENDER (a Role), and MALE (a Concept)

MAN $\equiv$ (CMeet PERSON (CRestriction GENDER MALE))
$\equiv \lambda x.$ (PERSON x) & $\forall y$ ((GENDER x y) => (MALE y))

This has the familiar reading

$\forall x$ (MAN x) <=> ((PERSON x) &
$\forall y$ ((GENDER x y) => (MALE y)))

The gloss for this is "a MAN is a PERSON all of whose GENDERs are MALE".

2.6 Classification in the S-component

The S-component organizes predicates in a taxonomy. This taxonomy is established on the basis of definitions given with the various S-component

constructors. For example, the taxonomy corresponding to my definitions of MAN, PARENT, and FATHER can be visualized as in Figure 1.

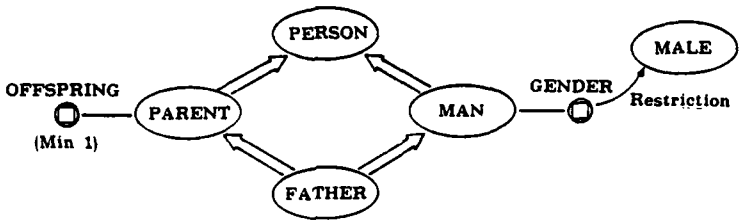


FIG. 1 A TAXONOMY OF PREDICATES

In the graphical notation of Figure 1, ellipses correspond to Concepts, and circled boxes correspond to Roles. The double-arrows denote *subsumption*, which is the basic taxonomic relation between predicates. The nature of subsumption is defined semantically. A predicate P is subsumed by a predicate Q if the meaning of P, as defined using the S-component constructors, includes (or entails) the meaning of Q. For example, the FATHER predicate I defined above is subsumed by the PARENT predicate, since by my definition the meaning of FATHER includes (or entails) the meaning of PARENT.

The subsumption relation between two predicates P and Q can also be understood as a universally quantified sentence. If P is subsumed by Q, then

$$\forall x\ (P\ x) \Rightarrow (Q\ x)$$

In the case of the FATHER predicate, we can read the fact that it is subsumed by the PARENT predicate as

$\forall$ x (FATHER x) => (PARENT x)

Predicate definitions are stored in the taxonomy by invoking a function called the classifier [9, 16]. The classifier is given the definition of a predicate *P*, and determines all of the subsumption relations between *P* and other predicates in the taxonomy. Determining these relations effectively installs the new predicate in the taxonomy. For example, when I defined FATHER as (CMeet PARENT MAN), the classifier determined the subsumption between FATHER and its subsumers PARENT and MAN. An important feature of the classifier is that it recognizes when a new predicate definition is identical to a definition given earlier to some other predicate. In this case, the classifier "merges" the new definition with the earlier one. essentially, it discards the new definition and returns the predicate associated with the original definition. For example, say I had defined FATHER as above and I now tried to classify some new predicate IS/A/DAD to which I had given the definition (CMeet PARENT MAN). The classifier would recognize that the definition of IS/A/DAD is identical to that of FATHER, discard the new definition, and return FATHER instead.

This notion of "merging" is significant in that it allows one to view the classifier as performing a recognition task. Given a predicate definition, the classifier can recognize whether that definition has already been used in specifying the meaning of a predicate. This ability to do recognition is a crucial part of how the S-component of KL-TWO provides quantification for the P-component.

2.7 Interfacing the P-component and S-component

The P-component of KL-TWO contains propositions, and the S-component of KL-TWO contains predicate definitions As I mentioned earlier, these definitions correspond to universally quantified sentences To understand the interface between the two components, it is helpful to view the S-component as encoding these sentences Given this, the job of the interface is to decide which of the S-component sentences should be instantiated in the P-component, thereby producing new P-component propositions

The approach taken to do this starts with the observation that propositions in the database are nothing more than the application of predicates to individuals. The database can thus be organized by associating with each individual *I* in the database the collection of predicates that have been applied to *I*. These groupings of predications serve as the locus of the interface between the P-component and the S-component. the task performed by the interface is to combine the predications applied to any individual *I* in a way that can be recognized by the S-component. The S-component then fetches the one quantified sentence most applicable in this situation, and the interface instantiates this sentence in the P-component. This in turn produces new predications of *I*.

How is this "predicate combination" and "quantified sentence fetch" performed? The predicate combination is done by abstracting from the predications of *I* a structured predicate that corresponds to the conjunction of the predications. This structured predicate can be given a definition with the S-component constructors

(CMeet, CMin, and others). The quantified sentence fetch is performed by invoking the S-component classifier on this definition.

The interface is best illustrated with an example. First, assume that the S-component contains the definitions of the predicates PARENT, MAN, and FATHER that I gave earlier -- these definitions correspond to the taxonomy of Figure 1. Next, let's say that I asserted in the P-component the following predications of the individual John

(PERSON John)
(OFFSPRING John Mary)

The interface groups together these predications of John and abstracts them into the structured predicate

λ x. ((PERSON x) & $\exists$ y (OFFSPRING x y))

This abstraction is valid because applying the structured predicate to John yields

((PERSON John) & $\exists$ y (OFFSPRING John y))

This expression is true by virtue of the predications I had asserted earlier. The interface encodes the abstract structured predicate using the S-component constructors CMeet and CMin, producing an expression that corresponds to the structured predicate

(CMeet PERSON (CMin OFFSPRING 1))

With the construction of this expression the interface has completed the "predicate combination" step.

The interface now starts the "quantified sentence fetch" step by invoking the S-

component classifier on the expression it has just constructed. The classifier attempts to recognize the relation between the predicate encoded by the expression and other predicates in the S-component taxonomy. In this case the recognition task is straightforward, since the predicate expression being classified exactly matches the definition of a predicate in the taxonomy, the predicate PARENT. The classifier performs this match, and in so doing it has essentially retrieved from the taxonomy the definition

PARENT $\equiv \lambda$x. (PERSON x) & $\exists$ y (OFFSPRING x y)

The interface reads this definition as the quantified sentence

$\forall$ x (PARENT x) <=> (PERSON x) & $\exists$ y (OFFSPRING x y)

This completes the "quantified sentence fetch". At this point all that is left for the interface to do is to instantiate the sentence it fetched in the P-component. It does this by asserting

(PERSON John) & (OFFSPRING John Mary) => (PARENT John)

From this the P-component will deduce (PARENT John).

2.8 Further Discussion

In the preceding section I covered some of the key ideas of the interface between the two components of KL-TWO. There are other significant aspects of the interface which I will only mention here briefly. For one, the "predicate composition" and "quantified sentence fetch" processes can be made incremental and very efficient. This in turn permits the interface to be run using a tight forward-chaining control structure. Briefly, whenever a new predication is asserted of an individual *I*, the "composition" and "sentence fetch" operations are immediately performed on the set of

predications of *I*. Since the operations are efficient, and since the inferences they perform are generally useful, the system incurs little overhead from using this control structure (see [19] for more details).

Additionally, the S-component can be used in answering queries to the P-component. For example, if (MAN John) were asserted in the P-component, then the S-component predicate definitions I gave earlier could be used to answer the query "is (PERSON John) true" (again, see [19] for details).

Before closing I would like to compare KL-TWO with another KL-ONE descendant, the KRYPTON system. KL-TWO and KRYPTON have several similarities, the principal one being that both are hybrid systems that contain a terminological component and an assertional component (in KRYPTON these are called the Tbox and Abox respectively). The two terminological components are roughly comparable, but the assertional components are substantially different -- the KRYPTON Abox is a full first order theorem prover. This difference in the nature of the two assertional components is reflected in how they're interfaced to their respective terminological components. The KRYPTON Abox is a theorem prover and hence doesn't require being supplemented by the Tbox to provide the system as a whole with quantification. Instead, in KRYPTON the Tbox is used simply for its ability to define the meanings of predicates. This is enforced by having the Tbox augment the unification algorithm of the Abox theorem prover [6].

2.9 The First Step

The current KL-TWO system is a first step towards providing a general efficient knowledge representation system. KL-TWO has good facilities for propositional deduction and for a class of quantified reasoning. Other kinds of reasoning (including much of the quantified reasoning that could be provided by a theorem prover) can not be performed by the various KL-TWO subcomponents. These forms of reasoning can only be captured at present by using hooks into the system, in particular demons. This restriction isn't necessarily a fatal shortcoming. KL-TWO isn't intended to provide the generality of a full first order language. This would require a full first order theorem prover, and much as these programs have improved recently [17], they are still prone to computational inefficiency.

KL-TWO focuses instead on providing an integrated collection of useful representation facilities that are well-motivated semantically and have efficient inference algorithms. I have already mentioned that for KL-TWO, the S-component and P-component are just a beginning. The system was designed so that it could be extended gracefully. My colleagues and I are investigating at this time a number of possible extensions. As our investigations mature and as we understand how to perform other forms of reasoning, we can incorporate new components into KL-TWO, thereby extending the system's generality and its usefulness to AI research.

2.10 Acknowledgements

To those that have contributed in one way or another to this research, I offer my heartfelt thanks. They are Dave McAllester, who provided me with RUP; Rusty

Bobrow, David Israel, Tom Lipkis, Jim Schmolze, and Dave Stallard, my KL-TWO cohorts, and Brad Goodman and Suzie Weaver, who both read many drafts

REFERENCES

[1] Brachman, R.J.
On the Epistemological Status of Semantic Networks.
In Findler, Nicholas V. (editor), *Associative Networks - The Representation and Use of Knowledge in Computers*. Academic Press, New York, 1979.

[2] Brachman, R.J.
An Introduction to KL-ONE.
In Brachman, R.J., et al. (editors), *Research in Natural Language Understanding. Annual Report (1 Sept. 78 - 31 Aug. 79)*, pages 13-46. Bolt Beranek and Newman Inc., Cambridge, MA, 1979.

[3] Brachman, R.J. and Schmolze, J.
An Overview of the KL-ONE Knowledge Representation System.
Cognitive Science, 1984.

[4] Brachman, R. J., and Levesque, H. J.
Assertions in KL-ONE.
In Schmolze, J. G., and Brachman, R. J. (editors), *Proceedings of the 1981 KL-ONE Workshop*, pages 8-17. Bolt Beranek and Newman Inc. Report No. 4842, 1982.

[5] Brachman, R.J., Fikes, R.E., and Levesque, H.J.
KRYPTON: A Functional Approach to Knowledge Representation.
IEEE Computer 16(10), October, 1983.
Also available as Fairchild Technical Report No. 639 or as FLAIR Technical Report No. 16.

[6] Brachman, R. J., Fikes, R. E., and Levesque, H. J.
KRYPTON: Integrating Terminology and Assertion.
In *Proceedings of the Third National Conference on Artificial Intelligence*, pages 31-35. The American Association for Artificial Intelligence, Washington, D.C., August, 1983.

[7] Doyle, J.
Truth Maintenance Systems for Problem Solving.
AI Technical Report 419, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, January, 1978.

[8] Israel, D.J.
Interpreting Network Formalisms.
Computers and Mathematics with Applications 9(1):1-13, June, 1983.
Special Issue on Computational Linguistics - Nick Cercone, guest editor.

[9] Lipkis, T. A.
A KL-ONE Classifier.
In Schmolze, J. G., and Brachman, R. J. (editors), *Proceedings of the 1981 KL-ONE Workshop*, pages 128-145. Bolt Beranek and Newman Inc. Report No. 4842, June, 1982.

[10] McAllester, D. A.
An Outlook on Truth Maintenance.
AI Memo 551, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, August, 1980.

[11] McAllester, D. A.
Reasoning Utility Package User's Manual.
AI Memo 667, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, April, 1982.

[12] Moser, M. G.
An Overview of NIKL, the New Implementation of KL-ONE.
In Sidner, C. L., et al. (editors), *Research in Knowledge Representation for Natural Language Understanding - Annual Report, 1 September 1982 - 31 August 1983*, pages 7-26. BBN Laboratories Report No. 5421, 1983.

[13] Rich, C.
Knowledge Representation languages and the Predicate Calculus: How to Have Your Cake and Eat It Too.
In *Proceedings of the Second National Conference on Artificial Intelligence*, pages 193-196. The American Association for Artificial Intelligence, Pittsburgh, PA, August, 1982.

[14] Schmolze, J. G., and Brachman, R. J.
Summary of the KL-ONE Language.
In Schmolze, J. G., and Brachman, R. J. (editors), *Proceedings of the 1981 KL-ONE Workshop*, pages 233-260. Bolt Beranek and Newman Inc. Report No. 4842, Appendix D, 1982.

[15] Schmolze, J. G., and Israel, D. J.
KL-ONE: Semantics and Classification.
In Sidner, C.L., et al. (editors), *Research in Knowledge Representation for Natural Language Understanding - Annual Report, 1 September 1982 - 31 August 1983*, pages 27-39. BBN Laboratories Report No. 5421, 1983.

[16] Schmolze, J.G., Lipkis, T.A.
Classification in the KL-ONE Knowledge Representation System.
In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, 1983.

[17] Stickel, M. E.
A Nonclausal Connection-Graph Resolution Theorem-proving Program.
In *Proceedings of the Second National Conference on Artificial Intelligence*, pages 229-233. The American Association for Artificial Intelligence, Pittsburgh, PA, August, 1982.

[18] Vilain, M. B.
A System for Reasoning about Time
In *Proceedings of the Second National Conference on Artificial Intelligence*, pages 197–201. The American Association for Artificial Intelligence, Pittsburgh, PA, August, 1982

[19] Vilain, M. B., and McAllester, D. A.
Assertions in NIKL.
In Sidner, C. L., et al. (editors), *Research in Knowledge Representation for Natural Language Understanding – Annual Report, 1 September 1982 – 31 August 1983*, pages 45–80. BBN Laboratories Report No. 5421, 1983.

3. THE NIKL PROGRAMMING ENVIRONMENT

David Stallard

3.1 Introduction

There are two principles that can be observed from the history of attempts to make computers useful for programming. The first principle is the importance of a formalism in which to express oneself, while the second is the inadequacy of that formalism alone for getting day to day work done. It is this inadequacy which forces us to make the distinction between a language and a medium in which we use the language. Thus in the beginning of the computer field FORTRAN was developed as a high-level formalism for the representation of algorithms. Later it was found that this was not sufficient in itself, and such tools as editors and debugging facilities were developed. In time these came to be seen as necessary, and indeed as keys to reasonable levels of productivity.

We have found this paradigm to apply equally well in the case of AI languages for knowledge representation. Here the "programming problem" is building a taxonomic lattice of descriptions instead of a system of procedures. Nonetheless, the activity is still more complicated and more subject to error and backtracking than first anticipated. Real domains always reveal hidden complexities when one attempts to model them. One doesn't want to have to, and indeed can't, write down a perfect description of a domain the first time, and enter it into the system all at once. It would be much better to create a rough working version, and then to be able to modify it incrementally as often as one needs. This activity should be as efficient as

possible for the user, so that he may concentrate on the more subtle problems of logical description.

In this paper we will discuss the programming environment for the NIKL knowledge representation system. We use the term "programming environment" instead of "tools" so as to make a distinction between what we feel are three levels of usefulness: the first, a formalism without tools, second a formalism with a set of tools, the third, a system in which the formalism and the tools are integrated together. Our programming environment has many capabilities for creating concepts and roles which interact with and support one another. These include defining, deleting, editing, prettyprinting, dumping definitions onto files, and noticing changes to definitions on a file, and examining the lattice with a "browser". It is both inspired by and embedded in the language INTERLISP [2], itself an excellent example of a helpful programming environment.

3.2 Basics: Defining, Editing, and Printing

The core of the NIKL programming environment is the non-procedural language CKLONE, which allows one to specify roles and concepts. The following is an example of a concept specification.

```
(DEFCONCEPT TANKER (SPECIALIZES SHIP)
                   (RES CARGO (VRCONCEPT OIL)))
```

The user has only to type this expression to his terminal. It creates and returns as its value a concept "TANKER" that specializes the concept "SHIP". The concept "TANKER" restricts the filler of the role "CARGO" to be the concept "OIL". Although it is not shown in the example, the restriction clause RES can include a MAX and a MIN

clause for specifying the number of fillers the role may have at a concept, or a NUMBER clause when MAX and MIN are the same. Another token could have been included at the top level of the DEFCONCEPT form that would have made TANKER a "natural kind", this is the atom PRIMITIVE. This would prevent anything that does not make explicit reference to TANKER being classified as one.

The definition forms can accept forward references without difficulty. That is, if there had been no concept "OIL" at the time the above DEFCONCEPT form was evaluated, one would have been automatically created as a subc of THING and given the name "OIL". The user would have been notified of this with a printed message. He could then have added features to "OIL" at his leisure. Since a given set of role and concept definitions are often mutually dependent this feature is frequently useful, we have dubbed it "FOMing" (for Find Or Make).

There is a similar construct called DEFROLE for creating roles. It enables the user to specify domain and range. For example, one might define the role "CARGO" to be

```
(DEFROLE CARGO (DIFFERENTIATES TEMPORARY-SUBPART)
               (DOMAIN VEHICLE)
               (RANGE MATERIAL))
```

This creates a role which is a differentiation of the role TEMPORARY-SUBPART, its domain is the concept VEHICLE and its range is the concept MATERIAL. In what follows I will most often use DEFCONCEPT as an example, but the reader should remember that most of what is said applies to DEFROLE as well.

It is important to remember that names such as "SHIP" and "CARGO" have no

essential meaning in themselves, nor are they a part of the NIKL formalism. Names are mapped to the concepts or roles they denote through a "namespace" mechanism that is external to the taxonomy. This mapping is not made directly from symbol to object. Rather, it is made first between the name and a data structure called a "term". This data structure has pointers to the role or concept object denoted, to the definition as entered by DEFCONCEPT or DEFROLE, and to a list of concept and role names that referenced the term in their own definitions. We call the latter the "dependents" of the term. Note that a given name can be used to denote both a concept and a role. Thus the notion of "term" is not identical with "name"; it must be regarded as an ordered pair of name and type. In our example above there would be a concept term "SHIP" which would list as one of its dependents the concept term "TANKER". There would also be a role term "OIL" which would list "TANKER" as one of its concept dependents. The term for TANKER would have a pointer to a concept and a pointer to exactly the definition above. If we later add another concept "SUPER-TANKER" as a specialization of TANKER, the term for TANKER would list as one of its dependents the concept SUPER-TANKER.

Now suppose we want to change the definition of the concept "SHIP", perhaps to add the role DISPLACEMENT to the definition.

```
(DEFCONCEPT SHIP (SPECIALIZES VEHICLE)
                 (RES LENGTH (NUMBER 1) (VRCONCEPT METERS))
                 (RES WEIGHT (NUMBER 1) (VRCONCEPT TONS))
                 (RES CARGO (NUMBER 1) (VRCONCEPT MATERIAL)))
```

We call the function EDITCONCEPT on SHIP. It goes to the term data structure for SHIP and fetches the definition form above. It passes this to the INTERLISP structure editor, and the user many then edit the definition as he pleases, treating the definition as list structure and using the standard editor commands. In this case the user might type the editor command:

```
(N (RES DISPLACEMENT (NUMBER 1) (VRCONCEPT TONS)))
```

The "N" command adds its argument form to the end of the list that is the "current expression" of the editor. This current expression may be changed by various FIND and motion commands. The user may also use a replace command, R, to replace one expression with another. Thus one could for example change the the VRCONCEPT of the restriction on WEIGHT to be KILOGRAMS instead of tons. When the user is satisfied with the definition he exits the structure editor by typing "OK", and EDITCONCEPT calls EVAL on the new definition form.

It is the code for DEFCONCEPT itself that handles all the modifications that must be made. In this way the several ways that a concept may be redefined – through editing, through typing in a new DEFCONCEPT form or through loading it from a file – may all be handled uniformly. Thus, what follows can be taken as describing the mechanism behind all three of these cases.

DEFCONCEPT first checks the definition for any syntax or other errors. It is particularly important to prevent the user from making certain semantic errors, for example restricting a role at a concept above its domain. If any errors are found, the appropriate message is printed and control is returned to EDITCONCEPT. The user is then allowed to either continue the edit to correct the problem or abort the edit by typing "STOP". This is a generally useful feature, if at any point during the session the user exits the editor in this way no changes are made to either definitions or objects.

If the new definition is approved, it is substituted for the old in the term data

structure for SHIP. DEFCONCEPT creates and substitutes into the term a new concept object that restricts the DISPLACEMENT role, as well as the LENGTH, WEIGHT, and CARGO roles it restricted previously. DEFCONCEPT then notes that SHIP has TANKER as a dependent, and through transitivity SUPER-TANKER as well. It rebuilds these from their definitions, using as a parent object the new concept object for SHIP that includes the DISPLACEMENT role. The system also rebuilds those concept terms inferred by the classifier to have SHIP as their parent. The new objects produced are substituted for the old in the terms for TANKER and SUPER-TANKER. Finally, DEFCONCEPT adds the concept term SHIP to the dependents for the role term DISPLACEMENT and removes the old concept objects from the system, reclassifying the new objects if the old objects were classified. The message "(Concept SHIP redefined)" is printed.

We can also "prettyprint" concepts and roles. This is different from simply examining their definitions, since NIKL may have made certain inferences about the object - such as restrictions inherited from super concepts - that are not present in the definition. We can see that TANKER has inherited the new role DISPLACEMENT by typing "PPC TANKER T" where the extra argument "T" tells the prettyprint command PPC to show all inheritance. This would print.

```
Concept TANKER
    Type: Generic
    Parents: SHIP
    Children: SUPER-TANKER
    Restricts:
        CARGO
            VR = OIL
            #R = (1 1)
        LENGTH
            VR = METERS
            #R = (1 1)
        WEIGHT
            VR = TONS
```

```
    #R = (1 1)
DISPLACEMENT
    VR = TONS
    #R = (1 1)
```

A similar command, PPR, is included for prettyprinting roles.

3.3 Dumping Definitions onto Files

The system includes role and concept definitions as INTERLISP file package types, thereby allowing us to put role and concept definitions onto a symbolic file. To use this facility we call the INTERLISP function FILES?, typing the form (FILES?) to the keyboard. This function tells us which role and concept definitions have not yet been put on any file, as well as which files now need to be dumped because of changes to the definitions on them. For example, the concepts SHIP, TANKER and SUPER-TANKER, together with the roles they use, may reside on the file SHIP-DESCRIPTIONS. After the edit above, the function FILES? would tell us that SHIP-DESCRIPTIONS needed to be re-written. If we had in the meantime defined another concept FREIGHTER, but had not yet placed it on a file, FILES? would tell us so. It would ask if we wanted to place the new definition on a file, and if the answer was yes, which file. We could then tell the system either to add the definition to SHIP-DESCRIPTIONS or to put it on some new file.

The FILES? function does not actually make any changes to the disk. To write a file, either to update it after changes or additions or to write it for the first time, we simply use the standard INTERLISP function MAKEFILE, as in the expression (MAKEFILE 'SHIP-DESCRIPTIONS). This would place our definitions on a standard INTERLISP disk file.

If during a future session with the system we should want to use the definitions we placed on the file SHIP-DESCRIPTIONS, we would simply type (LOAD 'SHIP-DESCRIPTIONS). Each definition written on the file would be evaluated in turn, and appropriate role and concept objects would be built from them.

Occasionally we may misspell the name of a role or concept inside a definition form and not notice it at the time. The system does not attempt to do spelling correction on a name and instead simply assumes that the user means to define it later. The function CKLONEStatus is provided to tell the user which roles and concepts were created in this way ("FOMed") and not subsequently defined by the user in their own DEFCONCEPT or DEFROLE form. If the user wishes, it will provide him with the names of the roles and concepts that use this name inside their definition forms.

A number of other utilities are provided to help the user deal with definitions on files, these may be found in the last section of this report.

3.4 Examining the Taxonomy

Printing a single role or concept is of course not always sufficient. Often one wants to see part or all of the taxonomy. There are two programs for this, KLONEDRAW and the NIKL "browser". Since KLONEDRAW has been discussed in a previous report (see J. Schmolze's report in [1]) we will discuss the browser, which was developed by Tom Lipkis of ISI.

The browser will draw the concept or role lattice into a window in graph form, with the names of the objects as nodes and the lines between them denoting super or

differentiation links. The user can use the mouse to point to a particular object and bring up a menu of options that includes most of the functions of the programming environment - prettyprinting, editing, inspection of definition and the like.

The top level entries are NKBC (Browse Concepts) and NKBR (Browse Roles). Both are NLAMBDA NOSPREADS and the simplest use is to give them the name of a node to be used as a root. (E.g., *(NKBC Person)* will show all concepts below Person.) If no argument is supplied, the top node of the appropriate type is used (NKMostGeneral...).

More control over the contents and layout of the graph can be had by supplying any number of the following options as arguments

```
(Below Name1 ... NameN)  or  (B Name1 ... NameN)
        Include nodes below any of Name1 ... NameN.

(Above Name1 ... NameN)  or (A Name1 ... NameN)
        Include nodes above any of Name1 ... NameN.

(Not Name1 ... NameN)
        Remove nodes Name1 ... NameN.

(NotBelow Name1 ... NameN)  or  (NB Name1 ... NameN)
        Remove nodes below any of Name1 ... NameN.

(Vertical)
        Put the root at the top (as opposed to the left).

(Tree)  Force the graph to be a tree.  Nodes displayed more than once are
        boxed.

Name    Equivalent to (Below Name).
```

In addition to the NLAMBDAs NKBC and NKBR, which are meant to be typed in, the following LAMBDAs are available for use from programs.

```
(NKBrowse NodeType Options Window)
        NodeType is one of Concept or Role, and Options is a list of
        options as above.  If Window is supplied, it will be reused.
        The browser window is returned.


(NKReBrowse Window)
```

Redisplays the graph in the specified browser window (reflecting any changes to the taxonomy). Window can also be the atom Concept or Role, in which case the most recent browser window displaying the appropriate type of object is redisplayed (unless it has been closed). Window = NIL is the same as Window = Concept.

3.5 Miscellaneous

An additional feature of the NIKL programming environment is integration with the INTERLISP programmers assistant. This facility of INTERLISP enables the user to "UNDO" the effects of certain commands he types, like SETQ, DEFINEQ, etc. One can also UNDO an UNDO. In our system it is possible to UNDO some commands. For example, one can undo a concept definition by typing "UNDO DEFCONCEPT". If a superseded definition exists it will be restored. If one does not exist, the concept term will simply be removed from the system. If one then were to type "UNDO UNDO" the DEFCONCEPT just typed will be restored. This process can be repeated ad infinitum. The following is the current list of undoable functions. DEFCONCEPT, DEFROLE, EDITCONCEPT, EDITROLE, NKDELETE, NKRESTOREDEF.

The NIKL programming environment includes a number of other utilities. A partial list follows. Where "Type" is included as an argument, case variants of the words "role" and "concept" may be included. If the user does not supply a type, it is assumed to be "concept". All arguments are evaluated unless otherwise stated.

NKReInit() - re-initializes the entire NIKL system. All objects and definitions created by the user are thrown away, and the taxonomy is reset to the initial net of concepts and roles supplied by the system. Useful for starting over entirely.

NKCALLS(Name Type) - returns an assoc list of the role and concept terms that the specified term uses.

NKCALLERS(Name Type) - returns an assoc list of the role and concept terms

that use the specified term.

NKCHANGENAME(OldName NewName Type InTerm InTermType)

- changes the use of OldName of Type in InTerm of InTermType to be NewName. If InTerm is not supplied every term that uses it (via NKCALLERS) is so changed. Affected definitions are re-evaluated.

NKCOPYDEF(Name1 Name2 Type) - Makes Name2 of Type have a definition isomorphic to that of Name1 of Type. Error if no such definition exists.

NKDELETE(Name Type) - deletes Name of Type from the system. The object attached to Name will be removed from the taxonomy.

NKFINDCALLERS(Atoms Files) - analog of INTERLISP FINDCALLERS for NIKL Tells you which role and concept definitions on Files contain something in Atoms. Args may be either atoms (single) or lists.

NKEDITCALLERS(Atoms Files Coms) - analog of INTERLISP EDITCALLERS.

NKGETDEF(Name Type File) - returns the definition of Name as a Type. If File is given the definition will be fetched from it. Otherwise it will be looked for in the symbol table.

NKGETOBJ(Name Type) - returns the object attached to the term Name of Type.

NKLOADDEF(Name Type File Ldflg) - analog of INTERLISP LOADDEF. Gets and evaluates the specified definition from File.

NKRESTOREDEF(Name Type) - retrieves and restores the definition of Name as a a Type that was superseded by redefining. Prints a message if no such definition exists.

NKREBUILDTAXONOMY() - collects up all stored definitions, re-initializes the taxonomy using NKReInit, and then reevaluates the definitions.

NKRENAME(CurrentName DesiredName Type) - renames the term CurrentName of Type to be DesiredName. If no such term exists, or if DesiredName is already in use, the user is informed and no further action is taken.

CKLONEProfile() - prints out the names of user-settable switches, their default and current values, and a brief description of what they do.

CCC(Name) - Does not evaluate its argument. Returns the concept argument attached to Name.

RRR(Name) - The same for roles.

REFERENCES

1. Sidner, C.L., Bates, M., Bobrow, R., Goodman, B., Haas, A., Ingria, R., Israel, D., McAllester, D., Moser, M., Schmolze, J., Vilain, M. Research in Knowledge Representation for Natural Language Understanding – Annual Report, 1 September 1982 – 31 August 1983. Technical Report 5421, BBN Laboratories, Cambridge, MA, 1983.

2. Teitelman, W., Goodwin, J.W., Hartley, A.K., Lewis, D.C., Vittal, J.J., Yonke, M.D., Bobrow, D.G., Kaplan, R.M., Masinter, L.M., Sheil, B.A. *INTERLISP Reference Manual.* Revised October 1978 edition, Xerox Palo Alto Research Center, Palo Alto, CA, 1978.

4. PLANNING IN A CHANGING WORLD

Andrew Haas

4.1 The Problem

A robot acts by executing plans. It sends a plan to its execution routine, which issues commands to the effectors and so performs the actions described in the plan. Of course the execution routine can perform these actions only if the proper preconditions hold. Let us say that the robot executes a plan if it sends that plan to its execution routine, whether or not the execution routine manages to carry out the actions in the plan.

The task of a planner is to search for a plan such that if the robot were to execute that plan, its goals would be achieved. So it must reason about what would happen if the robot were to execute a given plan. Suppose the planner finds that if the robot were to execute plan P1 a disastrous event E1 would occur, but if it were to execute plan P2 a desirable event E2 would occur. The robot executes P2 and not P1, so E2 occurs and E1 does not occur. Yet E1 would have occurred if the robot had executed P1, so it is a possible event, but not an actual one. And the planner must recognize that it is possible so it can warn the robot not to execute P1. Thus planning raises a hard problem in knowledge representation to describe events that are possible but not actual.

Situation calculus is a theory of planning that solves this problem, but at a heavy price. It depends on the assumption that every situation persists until changed

by the robot, the robot's actions are the only cause of change. This paper presents a planning theory in which things can change without the robot changing them. Section 2 describes the assumptions of situation calculus, and how they lead to conclusions about what would happen if the robot were to execute a given plan. Section 3 lays down a new set of assumptions, which allow changes that are not caused by the robot's action, and shows how these assumptions lead to conclusions about what would happen if the robot were to execute a given plan. Section 4 shows how to formalize this reasoning using modal logic. Section 5 extends the theory and formalism to handle more than one agent, and 6 sketches an extension to handle beliefs and knowledge. The final section compares the results to some related work.

4.2 Situation Calculus

The following account combines the theory of [7] with the practice of planners like [2]. There are a set of possible situations and a set of actions. At every moment one of the situations holds, or the robot is performing one of the actions, but not both. Every action has a unique name, which is also the plan for performing that action. Suppose that whenever the world is in situation S and the robot executes the plan for action A, it performs action A. Then the robot is able to perform action A in situation S. If the robot performs action A it will change the situation, but the new situation is not uniquely determined by S and A. There are many situations that could result from performing action A in situation S. If the robot is able to perform action A in situation S, and S' is a possible result, we say that action A can turn S into S'. This relation is taken as primitive in situation calculus.

Suppose that the current situation is S, and the robot is able to perform action

A. Suppose further that for all S' such that A can turn S into S', block B is on block C in S'. Then if the robot were to execute the plan for action A, block B would be on C afterward. Since A is only a possible action, and S' only a possible situation, this does not mean that block B will actually be on block C. Thus situation calculus describes events that are possible but not actual by introducing possible situations and possible actions. (In fact situation calculus doesn't even include any way of asserting that a situation or action actually happened at a particular time. But it could be added.)

Situation calculus is a theory of planning, not a formalism for planning. To build a formalism one must represent the ideas of situation calculus in some formal language. One possibility is to use first order logic and include situations in the domain of discourse. Another is to use modal logic, with situations acting as possible worlds [13]. Whichever route you choose, it's important to distinguish the formalism from the theory it expresses.

Suppose the robot is not able to perform action A, but it executes the plan for action A anyway. What happens? Of course the robot will not perform action A, but the unsuccessful attempt may have side effects. Situation calculus says nothing about these side effects, and this is an important omission. Suppose the robot wants to cross a frozen river, but it doesn't know whether it is able to perform this action because it doesn't know whether the ice is thick enough to bear its weight. If the ice is too thin, and it executes the plan for walking, it will end up at the bottom of the river. Situation calculus cannot assert this, because it does not describe the effects of unsuccessful attempts to perform actions.

A situation is an unchanging state of the whole world. So if anything in the world is changing at time t, the world is not in any situation at time t. If the robot starts an action at time t, we can't use situation calculus to predict the result because there is no initial situation. Once we allow changes that aren't caused by the agent, we may as well admit that there is no time when nothing in the world is changing. Then the world is never in any situation. Thus there is no generalization of situation calculus to handle changes that aren't caused by the agent. We need some way to distinguish actual from possible events without using situations.

4.3 A New Theory of Planning

Time is a set of instants ordered like the real numbers. Every event takes place during an interval of time, it begins at some instant i and ends at a later instant j. A condition likewise holds during an interval. To make this a little clearer, imagine that at *each instant the world is in a state* (an idea that was used with great success in classical physics). Whether a particular event took place in the interval from i to j depends only on the states of the world at instants between i and j inclusive, and likewise for conditions.

These assumptions about time suffice for the present work. To develop the theory further, one would need to consider the problems of interval-based theories of time. For example, does it make sense to say that a condition holds at a single instant? Suppose my digital watch reads "12.00" throughout a one-minute interval starting at noon. It's tempting to say that it read "12.00" at *each* instant during that interval. But this interval is followed by another, during which my watch reads "12:01". Then what was the reading of my watch at the instant where these two intervals meet? Such questions are considered in [10] and [1].

If the world never changes until the robot acts, the robot can wait as long as it wants before starting to execute a plan. If the world is always changing it makes a big difference when the robot starts executing a plan. If you want to catch the 6:30 bus you have to leave the house by 6:25. This means that the planner must choose not only a plan but also an execution time, and the routine that executes plans must have some way to know when this time arrives.

For this purpose the robot has a clock, which divides time into a series of numbered intervals called ticks. For every whole number n, the n-th tick begins at the instant (clock n) and ends at (clock n+1). During this interval the reading of the clock is n. Whenever the robot records an event it uses the reading of the clock to indicate when the event occurred. For example, it might record that the doorbell rang during tick number 897.

If all ticks were the same length, the robot could use its clock to measure intervals accurately. I will not assume this. Of course one could build a robot with an accurate sense of time, but I would like to apply this theory to people, whose sense of time is only roughly accurate. You can't tell the difference between a minute and fifty seconds without a watch, though you can tell the difference between a minute and five seconds. So I will assume only that all ticks are roughly the same length - say between a second and half a second. Suppose the planner decides that to get to the train station by 6PM, the robot should start executing a certain plan before 5:30. At this point its internal clock reads 500, and the clock on the wall reads 5:25. If a tick takes less than a second, 300 ticks take less than 5 minutes, so the internal clock will reach 800 before 5:30. Therefore the planner tells the plan execution routine to start

execution when the internal clock reads 800. This is like a human looking up at the clock and saying "I'd better leave in a couple of minutes".

The planner's job is to reason about what would happen if the robot were to execute a given plan. It must prove statements like

1 If the robot were to start executing plan P now, event E would happen.

Our theory must describe the truth conditions of this proposition. This is a hard task, because the proposition is a counterfactual conditional. The difference between counterfactuals and other conditionals is illustrated by the following examples.

2 If Oswald didn't shoot Kennedy, somebody else did.

3 If Oswald hadn't shot Kennedy, somebody else would have.

The first sentence is an ordinary conditional, the second is a counterfactual. Their truth conditions must be different, because the first is not controversial and the second is.

According to various philosophers [6], when we assert (3) we are talking about certain possible worlds other than the actual world. In these worlds Oswald did not shoot Kennedy, but in other ways they are much like the actual world. (3) is true if somebody else shot Kennedy in all of these worlds. For example, suppose there was a conspiracy against Kennedy, with other assassins waiting to shoot him if Oswald failed. This conspiracy also existed in the possible worlds where Oswald didn't shoot Kennedy, so in those worlds one of the other assassins shot Kennedy. Thus (3) might be true if there had been a conspiracy against Kennedy.

It would be best to invent a general theory of counterfactuals and apply it the special case of counterfactuals in the form (1). This is very difficult, and I will treat only the special case. Suppose the "now" mentioned in (1) is instant i. Our task is then to choose a set of possible worlds in which the robot executes plan P at instant i, but which in other ways are like the actual world. Let us call this set of worlds W(P,i). The problem is then, in what ways are the worlds in W(P,i) like the actual world?

Suppose it is now instant i, and event E has already happened. Then it is too late to do anything about event E. No matter what plan the robot were to execute at instant i, E would still have happened. We can ensure that this follows from our theory by postulating that all the worlds in W(P,i) have the same state as the actual world at every instant up to and including i. Then if it is now instant i, and event E has happened in the actual world, it happened in all the worlds in W(P,i) too. Then no matter what plan the robot were to start executing now, event E would still have happened.

One might expect that all the general laws that govern the actual world would also hold in the worlds of W(P,i), but this cannot be. Whenever the robot sends a plan P and an integer n to its execution routine, it executes P starting at (clock n). This is a general law governing the robot's actions, let us call it the Decision Law. Suppose that in the actual world the robot has told its execution routine to execute plan Q at instant i. This event occurred before instant i, so it happened in all worlds of W(P,i). If the Decision Law holds in all these worlds, the robot will execute plan Q at instant i in all these worlds. This simple robot can only execute one plan at a time, so if P is

distinct from Q the robot will not execute P at i in any of the worlds in W(P,i). This contradicts the requirement that the robot executes plan P at instant i in every world of W(P,i).

This is a good example of why counterfactuals are hard to analyze. A theory of counterfactuals must allow possible worlds in which there are exceptions to general laws. This is true for all sorts of counterfactuals, not just the ones about robots and their actions. If an eclipse has been predicted for tomorrow, it is physically impossible for that eclipse not to occur. There are no forces in our solar system that could disturb the moon's orbit that violently. Yet it's quite reasonable to say "If there were no eclipse tomorrow, a lot of people would be disappointed". By the same token I may know that it is physically impossible for the robot to execute plan P at instant i, given its hardware and programs. It is still quite reasonable to say "If the robot were to execute plan P at instant i, it would put block B on block C".

The Decision Law is not a fundamental law of nature. It is only a consequence of the robot's structure and the physical laws that govern its hardware. Still, let us simplify by assuming that the Decision Law is primitive, and it is the only law that governs what plans the robot executes. This means that if we suspend the Decision Law, the robot's execution of plans is not controlled by any causal law. The robot might execute any plan at any time. So let us look at the class of worlds in which all the general laws hold except the Decision Law. For any plan P, there are worlds in this class that have the same states as the actual world up to instant i, in which the robot starts executing plan P at instant i. By suspending the Decision Law we make room for the possible worlds we need.

In a general theory of counterfactuals one would show from general principles that there must be exceptions to the Decision Law, rather than simply postulating them. These exceptions would be kept to the minimum needed to make room for worlds in which the robot executes plan P at instant i. Far from minimizing the exceptions, I have thrown the Decision Law out the window (except that it still holds in the actual world). So my theory is only a rough approximation to the possible worlds theory of counterfactuals.

Then the worlds in W(P,i) resemble the actual world in two ways.

5 The state at every instant up to i is the same as in the actual world.

6 The general laws are the same as in the actual world, except for the Decision Law.

W(P,i) is the set of all worlds satisfying these conditions in which the robot executes plan P starting at instant i.

Conditions (5) and (6) are so weak that they place no restrictions on what plans the robot executes after instant i. If an event happens in all the worlds that satisfy these conditions, then it would happen no matter what plans the robot were to execute after instant i. That is, at instant i this event is bound to happen no matter what the robot does.

Let us consider a simple blocks world example. Suppose that if the robot executes the plan for picking up block B, and block B is clear, the robot successfully performs the action of picking up block B. If the robot picks up block B, it is holding block B when this action ends. These are general laws, and they hold in all the worlds in W(P,i) *for every* P *and* i. Assume that block B is clear in the interval between

I0 and I1. We should be able to show that if the robot were to execute the plan for picking up block B at instant I1, it would pick up the block successfully and it would be holding it when the action ended. This means showing that these events happen in every world in W(P,i), where P is the plan for picking up block B and i equals I1. Let W be any such world. Block B was clear from I0 to I1 in W, by condition (5). The robot executes the plan for picking up B in world W, and both general laws hold in W, so in world W the robot picks up block B and is holding it when this action ends.

This example only shows that we can still do the reasoning we could do in situation calculus. Let us consider an example that involves changes that aren't caused by the robot. Suppose that at instant I0 the robot is standing at point R on a railroad track. The train is going to reach point R in one minute, and the robot has no way to prevent this. That is, it happens in all worlds that satisfy (5) and (6) for i = I0. There are two plans available. If the robot executes the plan StandFast at an instant i, it remains in its current position for the next minute. If it executes the plan StepAside, it will move 6 feet to the left within the next minute. We should be able to prove that the robot can prevent itself from being hit by the train at I1 by executing StepAside at I0.

It doesn't make sense to say that somebody prevented an event by performing an action if that event was already impossible when the agent did the action [10]. If I lock the door of my office I can be quite sure that no tigers will enter. But I haven't prevented tigers from entering my office, because it was impossible in the first place (or at least very unlikely). Then to show that the robot can prevent itself from being hit by the train, we must show two things: at I0 it is possible for the robot to be hit

by the train, but if the robot were to execute StepAside at I0, it would not be hit by the train.

Suppose the robot is not hit by the train in any world that satisfies (5) and (6) for i = I0. Then at I0 it was already impossible for the robot to be hit by the train, and there was nothing to prevent. To see that this is not the case, recall that there are worlds in which the robot executes StandFast starting at I0. In these worlds the robot is still standing at R when the train arrives. On the other hand, there are worlds in which the robot executes StepAside at I0. In these worlds the robot is standing 6 feet to the left of the track at instant I1, and it doesn't get hit by the train.

In situation calculus the effect of executing a plan is to change the situation. In the new theory possible worlds replace possible situations. At an instant i the past is fixed, but there are still many possible futures. The effect of executing a plan P at instant i is to pick out a subset of these possible futures. Thus the robot has partial control of which possible future becomes actual

4.4 Formal Treatment

I have now presented a theory of possible events that uses possible worlds rather than possible situations, and so avoids the assumption that nothing changes when the robot is not acting. The next step is to represent the ideas of the theory in a formal language. If a sentence of a formal language represents a proposition, it is true in the intended model if and only if the proposition is true. So I must give the syntax of a formal language, define truth conditions, and describe the intended model.

Let us start with a first-order language. I introduce special variables that range over instants, and special constants that denote instants. Instant variables begin with "i", "j" or "k" and instant constants with "I", "J" or "K". For describing events and conditions that change over time I use special predicates called temporal predicates. The last two arguments of a temporal predicate are instants i and j, with i before j. The temporal predicate describes an event that happened, or a condition that held, during the interval from i to j inclusive. Thus (exec P i j) means that the robot executes plan P starting at instant i and ending at j.

The theory talks about possible worlds, so the obvious way to formalize it is to use modal logic and introduce a necessity operator ranging over the set W(P,i). This operator would have to take P and i as arguments, along with a wff. Something like this happens in Pratt's dynamic logic [13], but dynamic logic is a complex formalism far removed from ordinary modal logic. I will use a tricky approach that leads to a simpler formalism.

Let us introduce a necessity operator that ranges over all worlds that satisfy (5) and (6). These conditions do not mention the plan P, so the operator needs only two arguments, a wff and an instant i. I write [i]q to indicate that q holds in every world that satisfies (5) and (6). I said above that if an event happens in all worlds that satisfy (5) and (6), then at instant i it was bound to happen no matter what the agent did. So [i]q means that at instant i the events described by q are inevitable, no matter what the robot does.

Recall from ordinary modal logic that the sentence

[](p -> q)

holds iff q holds in every world where p holds. Now the sentence

(some j. (exec P i j))

holds in just those worlds in which the robot executes plan P at instant i. Therefore

[i](some j. (exec P i j)) -> q

holds iff q holds in every world satisfying (5) and (6) in which the robot executes plan P at instant i. that is iff q holds in all the worlds of W(P,i). So this sentence says that the events or conditions described by q would happen if the robot were to execute plan P starting at instant i.

The formal language is a quantified modal logic. There is a set of possible worlds, including a distinguished world called the actual world, and a relation of accessibility over possible worlds. All worlds have the same domain of discourse; there are no objects that could have existed but don't. For describing time we have a distinguished subset of the domain, the set of instants. There are constants that denote instants and variables that range over instants. The instants are totally ordered by the predicate "<", whose extension is the same in all possible worlds. Thus all possible worlds have the same time-line.

There is one important difference between this system and a standard quantified modal logic. In the usual systems there is an accessibility relation R, which is a binary relation over the set of possible worlds. A sentence is necessary at a world W if it is true at all worlds W' such that (R W W'). In this system world W' is an alternative to world W if W and W' have the same states up to instant i and obey every general law but the Decision Law. Therefore the relation R has one more argument, which is an instant. (R i W W') means that W and W' have the same states at every instant up to i

and obey all general laws but the Decision Law. Then for any fixed i, the relation (R i w w') must be an equivalence relation. Also, if i < j and w and w' have the same states at all instants up to j, they must have the same states at all instants up to i. So if i < j and (R j w w'), (R i w w'). The necessity operator has two arguments: a wff and a term that denotes an instant. [i]p is true if p is true at every world that is an alternative to the actual world at instant i.

A signature consists of an infinite set V of ordinary variables, an infinite set V' of instant variables, a set C of constants, a set C' of instant constants, and for each non-negative n, a set P of n-adic predicate letters. Instant variables start with the letter "i", "j" or "k". A term is a variable or a constant, an instant term is an instant variable or an instant constant. Every signature contains the binary predicate letter "<".

A model for a signature S is a 6-tuple <D, I, K, R, A, G>. D is a non-empty domain, I a non-empty subset of D (the set of instants), K a non-empty set of possible worlds, R a subset of I x K x K (the accessibility relation), and A an element of K (the actual world). The function G assigns an extension to every symbol in S. In particular, it assigns:

- To each x in V or C an element G(x) of D.
- To each x in V' or C' an element G(x) of I.
- To each n-adic p in P and w in K a subset G(w p) of the set of n-tuples of elements of D (the extension of p at w).

Finally, a model must have the following properties:

- The extension of "<" is the same at all worlds, and it totally orders the set of instants.
- For all i in I, (R i w w') is an equivalence relation.
- For all i, j in I and w, w' in K, if i < j and (R i w w') then (R j w w').

Wffs are defined recursively as follows.

- If P is an n-adic predicate letter and t1 ... tn are terms, (P t1 ... tn) is a wff.
- If p and q are wffs and v is a variable, then ~p, p & q, and (all v p) are wffs.
- If p is a wff and t is an instant term, then [t]p is a wff.

M{x/d} is the model like M except that the variable x denotes d. M{w} is the model like M except that the actual world is w. The truth-value G(w M) of a wff w in the model M = <D, I, K, R, A, G> is defined inductively as follows.

- G(p(t1 ... tn) M) = T if <G(t1)...G(tn)> is in G(A p), else F.
- G(~p M) = T if G(p M) = F, else F.
- G(p & q M) = T if G(p M) = T and G(q M) = T, else F.
- If x is in V, G((all x p) M) = T if for all d in D, G(p M{x/d}) = T, else F.
- If x is in V', G((all x p) M) = T if for all i in I, G(p M{x/i}) = T, else F.
- G([t]p M) = T if for all w in K such that R(G(t) A w'), G(p M{w}) = T, else F.

The first four clauses are just like the standard definitions for first-order logic, except that G takes an extra argument. The fifth clause takes care of quantification over instant variables, and the last one defines the necessity operator. Notice how this last clause resembles the clause for (all x p). The necessity operator is like a universal quantifier over possible worlds.

We define the existential quantifier and the modal operator <> for possibility in the usual way

(some x. P) <-> ~(all x. ~P)
<i>P <-> ~[i]~P

The inference rules of this system include the usual rules of first order logic, with the proviso that one can substitute only terms of type instant for a universal variable of type instant. The rules for the modal operator [i] are of two kinds, those borrowed from standard modal logic and those special to this system. Let us consider the first class. Whatever is true in all alternatives for instant i is true in the actual world. This is expressed by the axiom schema

[i]p -> p

Whatever can be proved from sentences that are necessary at i is itself necessary at i. It follows that if P is possible and Q is necessary, any sentence R that follows from P and Q is possible. For if R is not possible, ~R is necessary. ~R and Q entail ~P, so ~P is necessary, contradicting our hypothesis that P is possible.

There is one modal rule peculiar to this system. Any world that is an alternative for instant j is also an alternative for any instant i before j. So if a sentence is necessary at i (true in all alternatives for i) it is necessary also at j. This we capture with the schema

(i < j & [i]p) -> [j]p

These are all the inference rules needed to do the examples. With slightly stronger rules one can prove completeness.

I now present various axioms that are true in the intended model and useful for planning. General laws (except for the Decision Law) are supposed to hold in every possible world. So if sentence P expresses a general law, we have

(all i [i]P)

Of course the Decision Law holds in the actual world, and we can state this if we like.

W(P,i) is not supposed to be empty. That is, among the worlds that satisfy (5) and (6) there are some in which the agent executes plan P at instant i. So the sentence

<i>(some k. (execute P i k))

is true in the intended model.

If P is a temporal predicate, then its last two arguments must be instants i,j with i < j. If P = "on", for example, the sentence

(on x y i j) -> i < j

is true in the intended model. Since the truth value of (on x y i j) depends only on states between i and j inclusive, (on x y i j) is either true in all worlds accessible at j or false in all worlds accessible at j. If true it is necessarily true, and if false it is necessarily false. Thus the following are true in the intended model.

(on x y i j) -> [j](on x y i j)
~(on x y i j) -> [j]~(on x y i j)

With these axioms and inference rules we can formalize the arguments of section 2. Consider the blocks world example. Block B was clear from I0 to I1.

I0 (clear B I0 I1)

If block b is clear from i to j, and the robot executes (PickupPlan b) from j to k, it

picks up the block between j and k. This is a general law and holds in all possible worlds.

```
11 [i0](clear b i j) & (exec (PickupPlan b) j k)
       -> (PickUp b j k)
```

If the robot picks up block b from i to j it is holding block b at the end of this interval, that is from some k to j.

12 [i0](PickUp b i j) -> (some k (holding b k j))

Now assume that the robot executes (PickupPlan B) from I1 to I2.

13 (exec (PickupPlan B) I1 I2)

(10) through (13) entail

(some k. (holding B k I2))

Discharging the assumption (13) gives

(exec (PickupPlan B) I1 I2) -> (some k. (holding B k I2))

The remaining assumptions are all necessary at I1, so we can strengthen the conclusion to

[I1](exec (PickupPlan B) I1 I2) -> (some k. (holding B k I2))

And this is the desired theorem.

Now for the train example. I will formalize only the argument that it was possible at I0 for the robot to be hit by the train; the other half of the argument is just like the last example. At I0 the train is bound to reach point R at I1, no matter what the robot does.

14 [I0](at train R I1)

If the robot executes StandFast at I0, it will be at R at I1

15 [I0](some k. (exec StandFast I0 k)) -> (at robot R I1)

Now assume that the robot executes StandFast at I0.

16 (some k (exec StandFast I0 k))

(14) through (16) entail

17 (at robot R I1)

(14) and (15) are necessary at I0, and (16) is possible at I0, so (17) is possible at I0.

<I0>(at robot R I1)

This sentence says that it is still possible at instant I0 that the robot could be at point R at instant I1.

In situation calculus the only effect of an action is to create a new situation. In this formalism we describe the effects of executing plan P at instant i with sentences of the form

[i](some k (exec P i k)) -> q

where q can be any sentence of the language. This means that we can say a lot more about the effects of an action than we could in situation calculus. Suppose the robot rusts easily, and it wants to stay out of the rain at all times. We cannot describe this goal by describing a single situation, it is achieved only if the robot is out of the rain in all future situations. In the new formalism we can say that if the robot were to execute plan P at instant i, it would be out of the rain at all times after some future instant I.

[i](some k (exec P i k)) -> (all j. I < j -> ~(InRain j))

A more interesting possibility is that q could contain the necessity operator. If you are trying to give up smoking you might throw out all the cigarettes in the house, so that when your will weakens and you want a cigarette it will be impossible to get one. The sentence

[I1](all j. I1 < j -> ~(smoking j))

says that at I1 it is impossible for the agent to smoke any more. So the sentence

```
[I0]( (some j. (exec (ThrowOutPlan) I0 j))
      -> [I1](all j. I1 < j -> ~(smoking j))
    )
```

says that if the robot were to execute (ThrowOutPlan) at I0, then by I1 it would be impossible to smoke any more. Thus we can reason about a plan whose goal is to make it impossible to achieve another goal.

4.5 Multiple Agents

I now consider the case in which there is more than one agent. This means finding the truth conditions of a wider class of counterfactuals than were considered in section 3. We now consider all counterfactuals of the form "If agent A were to execute plan P at instant i, event E would happen". We must choose a set of possible worlds in which agent A executes plan P at instant i, but which are otherwise much like the actual world. Once again the question is, like the actual world in what ways?

Let us call this set of worlds W(A,P,i). As before the worlds in W(A,P,i) are like the actual world at all instants before i. Also, all general laws except the Decision Law hold in these worlds. Obviously the Decision Law cannot hold for agent A in these worlds. Otherwise we would get the same problems we had in the one-agent case. On the other hand, this doesn't necessarily mean that the Decision Law doesn't hold for other agents. Let us consider an example that bears on this question.

Suppose there are two robots, A and B. A heavy box is sitting on the floor. Neither robot is strong enough to lift the box, but if both robots lift at the same time they can put the box on the table. Suppose robot A has already decided to execute a

plan for lifting the box at I1. Then if robot B were to execute the same plan at I1, they would lift the box to the table. We must choose the set of worlds W(B,P,i) so that this counterfactual is true. And this means that the Decision Law must hold for agent A in these worlds. If not, the fact that agent A has decided to execute the lifting plan at I1 would have no bearing on whether A will execute the lifting plan at I1 or not. There would be worlds in W(B,P,i) at which robot A did not execute the lifting plan at I1. In those worlds the robots would not lift the box to the table. So by our account, it would be false that if robot B executed the lifting plan at I1, the robots would lift the box to the table.

So in the worlds of W(B,P,i), the Decision Law holds for every agent except B. This means that agent B can use predictions of other agent's behavior in his planning. He can also reason about the effects of his executing any plan P, even when the Decision Law says that he will not execute P. The situation is symmetric: in the worlds of W(A,P,i) the Decision Law holds for B but not for A, so that A can use predictions of B's behavior in his planning. As before, a general theory of counterfactuals would predict this, but I am content to postulate it.

So the worlds in W(A,P,i) resemble the actual world in two ways

17 The state at every instant up to i is the same as in the actual world.

18 The general laws are the same as in the actual world, except that the Decision Law does not hold for agent A.

W(A,P,i) is the set of all worlds satisfying (17) and (18) in which agent A executes plan P starting at instant i.

To formalize this I need a necessity operator with three arguments: an agent, an instant of time, and a wff. [A i]p means that p holds in all worlds that satisfy (17) and (18). That is, at instant i the events described by p are bound to happen no matter what agent A does. This does not mean that those events are bound to happen no matter what agent B does, for B may have the power to prevent things that are outside A's control. The accessibility relation now has four arguments: an agent, an instant and two worlds. (R A i w w') means that w and w' are alike up to instant i, and both worlds obey every general law except that the Decision Law does not hold for agent A. The formal construction of the language goes through as before, with obvious extensions.

There is one extension that is not obvious. In the one-agent system the Decision Law was only required to hold at the actual world. Now it is required to hold at the worlds in W(A,P,i) for every agent except A. We need an axiom that reflects this. The Decision Law says that if an agent sends plan P and integer n to his execution routine, the agent will execute plan P at (clock n). (plan A P n i) means that at instant i agent A sends plan P and integer n to his execution routine. Then we can state the Decision Law as follows.

(plan a p n i) -> (some k. (exec a p (clock n) k))

We must assert that at the worlds in W(A,P,i), the Decision Law holds for every agent b distinct from a. This can be written

```
[a i] a ≠ b ->
        ((plan b p n i0)
          -> (some k. (exec b p (clock n) k))
        )
```

On the other hand, for every plan p the worlds that satisfy (17) and (18) include worlds where agent a executes p starting at i.

<a i>(some k. (execute a p i k))

Thus we can extend the formalism to handle more than one agent.

4.6 Planning to Acquire and Use Knowledge

Most planners only plan to manipulate physical objects, like blocks. But many plans involve changing the robot's knowledge rather than changing the physical world. Suppose the robot wants to call Mary on the phone, and it has her number written on a piece of paper. It should plan to find out the number by looking at the paper and then dial it. Looking at the paper doesn't change any physical property of the world – it changes the robot's knowledge. To build such a plan the robot needs a theory of its own knowledge – just as it needs a theory of blocks to build plans for stacking blocks. But what theory?

If we want to write a program that believes that snow is white, we devise a knowledge representation in which we can represent that snow is white – for example, by writing "(white snow)". Then we add this sentence to a collection of sentences that are supposed to represent the program's beliefs. This practice suggests a theory: that beliefs are sentences of a knowledge representation language. This is the syntactic theory of belief. It can be extended to knowledge if we assume that knowledge is true belief (wrong, but close enough). McCarthy [9] and Moore and Hendrix [11] advocated this theory. Konolige [5] and Haas [3] applied it to planning.

It's natural to suppose that all the robot's beliefs are sentences of a single language L. Suppose the robot forms beliefs about its own beliefs – such as the belief that it knows what Mary's phone number is. Such a belief is a sentence of language L

that talks about sentences of L. Thus L must be its own meta-language. One can build such a language by starting with a first-order language and adding axiom schemas that describe the syntax, model theory and proof rules of that language (see [12] and [4]). In the same way the modal logic of this paper can be extended to a language that describes itself, and with this representation the planner can reason about its own beliefs.

In such a language one can describe the preconditions and effects of mental actions like perceiving, remembering and inferring. Consider the action of inferring q from p and p -> q by Modus Ponens. Its precondition is that the robot believes p and (p -> q), its effect is that the robot believes q. That is, if the robot believes p and and (p -> q) from instant i0 to i1, and it executes the plan (ApplyModusPonens p (p -> q)) from i1 to i2, then it will believe q from some instant i to i2. One can formalize this roughly as follows:

```
[i][(   (believe p i0 i1)
     & (believe (p -> q) i0 i1)
     & (execute (ApplyModusPonens p (p -> q)) i1 i2)
   ) -> (some i. (believe q i i2))
```

The variables p and q range over sentences of the robot's knowledge representation language. The notation is not precise, to make it so requires adding quotation marks. With axioms like this one can describe the preconditions and effects of actions like perception, introspection, memory retrieval, and inference, for details see (Haas 1983).

Often the robot needs to acquire knowledge in order to perform an action. It must know the digits of Mary's phone number in order to call her, for instance. The robot performs an action by executing a plan for that action, so it knows how to perform an action if it knows that some plan P is a plan for that action. From this

assumption, together with reasonable assumptions about the robot's effectors, we can show that the robot knows how to call Mary if it knows the digits of her phone number.

Suppose for simplicity the phone has push buttons instead of a dial. To "dial" Mary's phone number, the robot must find the right buttons on the phone and push them in the right order. The robot's hand is guided by its eye. If a plan specifies the position of an object in the robot's field of view, the execution routine can guide the hand to that object. Assume for simplicity that the robot uses Cartesian coordinates to identify positions in the field of view. Then if x and y are coordinates, (push x y) is a plan for pushing a button that appears at coordinates (x,y) in the robot's field of view. If the robot wants to push the button labeled "5", it must find that button at some coordinates (x, y) in its field of view and execute the plan (push x y). If planner knows that Mary's phone number is 5766, it can prove that the robot can dial Mary's phone number by executing a plan that looks like this:

```
Look at the buttons on the telephone.
Find the coordinates (x1,y1) of the button labeled "5".
(push x1 y1).
Find the coordinates (x2,y2) of the button labeled "7".
(push x2 y2).
Find the coordinates (x3,y3) of the button labeled "6".
(push x3 y3).
Find the coordinates (x4,y4) of the button labeled "6".
(push x4 y4).
```

If the planner doesn't know that Mary's number is 5766, it cannot prove that this plan is correct. So the planner must know the digits of Mary's number in order to find a plan for dialing her number.

For another example, suppose the goal is to ring a bell at exactly 7:00. Common sense suggests that without a clock, the planner cannot devise a plan for this goal,

and our theory predicts this too. The planner must tell the execution routine to start executing when the internal clock's reading is a certain number n. But the planner has no way of knowing what the reading on the internal clock will be at 7:00. If it had a clock, it could plan to watch the clock until it read 7:00, and then ring the bell. So our theory of planning yields reasonable predictions about what a robot has to know in order to find a plan for a goal.

4.7 Related Work

McCarthy and Hayes [8] tackled the problem of planning with events that aren't actions of the agent. My solution is based on the same intuitions as theirs, though I use very different formal devices. I will describe their solution (in the one-agent case) and the correspondence between the two.

Imagine two non-deterministic finite-state automata, one represents the robot and the other the external world. The outputs of the robot-automaton are connected to the inputs of the world-automaton, and vice versa. The robot can act on the external world by producing output, and the external world acts on the robot in the same way. Given the initial states of the two automata, their transition functions determine the possible histories of the system.

The robot can bring the external world into state S if there exists a sequence of outputs from the robot that would bring the external world into state S. That is, in all possible histories of the system in which the robot produces those outputs, the external world eventually reaches state S. Here we come to a problem. The robot's transition function may not permit it to produce those outputs, then there are no

possible histories in which the robot produces them. McCarthy and Hayes solved this problem by broadening their definition of a possible history of the system. They dropped the requirement that the robot produces the output dictated by its transition function, and instead allowed any output whatever from the robot.

This system closely resembles the intended model of my formalism. The robot-automaton corresponds to the programs that choose a plan to be executed, its output corresponds to the plans that are sent to the execution routine. The possible histories of the system correspond to the possible worlds. The transition function that governs the output of the robot-automaton corresponds to the Decision Law. Just as we had to allow possible worlds in which there were exceptions to the Decision Law, McCarthy and Hayes must allow possible histories in which the transition function does not govern the robot's output.

The basic idea of both systems is this. You draw a boundary between your robot and the external world, and describe the output that the robot sends across that boundary to the external world. Suppose that if the robot were to produce a certain output, it would achieve goal G, then the robot can achieve goal G. The trick is to place the boundary so that there is no question whether the robot can produce the output it wants, the only question is whether producing that output will cause the desired result. The robot can surely send any plan to its execution routine, so if the outputs are plans sent to the execution routine the robot can produce any output it wants.

McCarthy and Hayes claimed that there are many possible places to draw this

boundary, and none of them is best. I have drawn the boundary at a particular point, where the robot sends plans to its execution routine, and I claim that this choice is best. My chief argument is that it supports a simple and intuitive account of the knowledge needed for planning, as shown in the last section.

Thus my representation is derived from McCarthy and Hayes's in two steps. The first step is to decree that the robot's outputs are the plans it sends to its execution routine. The second step is to get rid of the finite-state automata, and take the possible histories of the system as possible worlds in a modal logic.

McCarthy and Hayes said that their formalism could also be used to represent certain counterfactuals. Thus they hinted at the idea that "the robot can achieve goal G by executing plan P" is equivalent to the counterfactual "if the robot were to execute plan P it would achieve goal G".

The intuition behind this theory comes from McCarthy and Hayes, but the formal methods come from McDermott [10]. He uses possible worlds (he calls them "chronicles") and he has the equivalent of my accessibility relation over worlds. There is one crucial difference. McDermott has no actual world. This makes it impossible to assert in his system that something actually happens, one can only assert that something could happen or must happen. He says that marking one world as actual would create paradoxes. If there is an actual world then something is inevitable if it happens in the actual world. This means that whatever happens was always inevitable, which is certainly a paradox. But why should we say that an event is inevitable if it happens in the actual world? Inevitability is a kind of necessity, and the basic idea of

modal logic is that an event is necessary if it happens in all possible worlds, actual or not. If we use this idea, as I have done, we can have an actual world with no paradoxes.

There is another difference between my formalism and McDermott's. McDermott does not use modal logic. Instead he uses first-order logic and includes possible worlds in his domain of discourse. He says (following Moore) that you can't build a good theorem prover for modal logic. I have seen no argument to support this claim, except that nobody has done it yet. I prefer to use modal logic because I find it more convenient than explicit mention of possible worlds. It is straightforward to translate my formalism into an equivalent set of first-order axioms.

Planning requires distinguishing between possible and actual events, which is a hard problem in knowledge representation. Situation calculus solves this problem, but it depends on the assumption that the world never changes unless the robot changes it. I have presented an alternative solution, which avoids this assumption by using possible worlds rather than possible situations. With extensions my theory can handle multiple agents and the role of knowledge in planning.

REFERENCES

1. Allen, James. A General Model of Action and Time. Department of Computer Science, University of Rochester, September, 1981

2. Fikes, R.E., and Nilsson, N.J. "STRIPS: A NewApproach to the Application of Theorem Proving to Problem Solving". *Artificial Intelligence* (1971), 189–208.

3. Haas, Andrew. *Mental States and Mental Actions in Planning*. Ph.D. Th., Department of Computer Science, University of Rochester, 1982.

4. Haas, Andrew. The Syntactic Theory of Belief and Knowledge. Bolt, Beranek and Newman, September, 1983.

5. Konolige, K. A First-Order Formalization of Knowledge and Action for a Multi-Agent Planning System. 232, SRI International, 1980

6. Lewis, David. *Counterfactuals*. Harvard University Press, Cambridge, Massachusetts, 1973

7. McCarthy, John. Situations, Actions and Causal Laws. 2. Stanford University A. I. Project, 1963

8. McCarthy, John and Hayes, Patrick. Some Philosophical Problems from the Standpoint of Artificial Intelligence. In *Machine Intelligence 4*. American Elsevier, 1969

9. McCarthy, John. First-Order Theories of Individual Concepts and Propositions. In *Machine Intelligence 9*. Halsted Press, New York, 1979, pp. 120–147.

10. McDermott, Drew. A Temporal Logic for Reasoning About Processes and Plans. 196. Department of Computer Science, Yale University, March, 1981.

11. Moore, Robert, and Hendrix, Gary. Computational Models of Belief and Semantics of Belief Sentences. 187. SRI International, Menlo Park, California, 1979.

12. Perlis, Donald. *Truth, Syntax and Reason*. Ph.D. Th., Department of Computer Science, University of Rochester, 1980.

13. Pratt, Vaughn. Semantical Considerations on Floyd-Hoare Logic. Proc. 17th IEEE Symposium on Foundations of Computer Science, 1976, pp. 109–112.

5. DOMAIN DEPENDENT SEMANTIC ACQUISITION

Margaret G. Moser

5.1 Introduction

This paper describes a method for acquiring the domain dependent semantic knowledge needed for a natural language processor. The method relies on user generated English examples of domain sentences, relieving the user of the need to know the underlying representation. IRACQ (for Interpretation Rule ACQuisition)[7] program to do this acquisition for IRUS, a natural language database retrieval system [2, 1] This paper begins with a simple example of IRACQ to illustrate features. Then, we define domain dependent semantic knowledge, and present an overview of IRUS and the implementation of IRACQ. After that, I present a description of another system, PSI-NIKL, and how the semantic acquisition could be done for it. Finally, there will be a brief discussion of domain porting and future work on IRACQ.

5.2 An Example of Semantic Acquisition

Semantic knowledge is used by the processor to (1) decide which phrases are meaningful, and (2) compute the meaning of those phrases. This is a recursive process Typically, a phrase, the matrix phrase, is composed of constituent phrases. The matrix is meaningful if all its constituents are meaningful and are combined in an acceptable way. The meaning of the matrix is computed from the constituent meanings.

---

[7] *The idea for IRACQ was originally suggested by Rusty Bobrow. The author wishes to thank him for helpful advice, and to thank Lyn Bates for valuable comments on earlier drafts of this paper.*

This section presents an example of how IRACQ acquires the knowledge needed to compute these interpretations In this example, IRACQ is going to acquire the semantic knowledge needed to interpret the verb "write", in the sense of writing papers and books More precisely, IRACQ will acquire the information needed to interpret phrases which refer to things in the WRITE-ACTION semantic class. A semantic class is some set of things in the domain. In the IRUS test domain, "write", "author" (when used as a verb), and "publish" are all used to talk about the same semantic class. The first step in the acquisition dialogue is for the user to enter an example sentence which we will call an exemplar

The dialogue comprising the rest of this section was obtained using the IRACQ system. Input from the user is shown in boldface, output from the system is in regular face, and comments are inserted in italics.

Enter an exemplar SENTENCE using WRITE: **JONES WROTE SOME ARTICLES**

IRACQ uses the full language processor of IRUS to parse the exemplar and interpret all its constituents[8] In this case, the last name "Jones" is interpreted as a reference to a member of the semantic class PERSONNEL.

JONES is in the semantic class PERSONNEL.
You can choose one of these generalizations: PERSON AGENT,
<CR> (to keep PERSONNEL),
or Exemplar (to give a new exemplar constituent): **PERSON**

IRACQ needs to know if the subject constituent must refer to a PERSONNEL member or to a member of one of the more general classes PERSON and AGENT. In case there was an error in the exemplar, e.g., the user mistakenly thought "Jones" was the name of an organization, the user may enter another exemplar constituent.

Selectional restrictions are one kind of domain dependent semantic knowledge. What kind of thing should each constituent be for the phrase to make sense? The possibilities are constrained to the semantic classes of the exemplar constituent. Now that IRACQ has elicited a selectional restriction on the subject of "write", it goes on to

---

[8]For now, assume IRUS has the semantic knowledge necessary for interpreting all the constituents. Uninterpretable constituents will be discussed in Section 5.7.

```
ARTICLE is in the semantic class BIBLIOGRAPHY.
You can choose one of these generalizations: WRITTEN-MATERIAL
                                             HOLDING,
<CR> (to keep BIBLIOGRAPHY),
or Exemplar (to give a new exemplar constituent): BIBLIOGRAPHY
```

Now IRACQ asks whether there are any prepositional phrases likely to modify "write".

```
Any Prepositional Phrases (PPs)? Yes
```

When additional constituents are possible, IRACQ will continue to prompt for more exemplars. Otherwise, the user would have to include all interpretable constituents in the first exemplar.

```
Enter an exemplar PP: IN NEWSWEEK

   Is the IN PP necessary for the meaning of WRITE? No
```

Sometimes, particular prepositions and PPs are mandatory to interpret a phrase. For example, in the IRUS test domain, "the title" is not interpretable but "the title of the book" is.

IRACQ now asks for synonyms for the preposition.

```
   What PREPs besides IN can be used this way: FOR

   NEWSWEEK is in the semantic class JOURNAL.
   You can choose one of these generalizations: WRITTEN-MATERIAL
                                                HOLDING,
   <CR> (to keep JOURNAL),
   or Exemplar (to give a new exemplar): JOURNAL

Enter an exemplar PP: OK
```

The "ok" signals the end of the user's specification of PPs, and IRACQ summarizes the information it has elicited.

```
To interpret: JONES writes ARTICLE in NEWSWEEK.
   requires: (JONES to be PERSON class)
             (ARTICLE to be BIBLIOGRAPHY class)
             (NEWSWEEK to be JOURNAL class)
```

For each constituent slot the user has specified, IRACQ uses its exemplar to create a "tag" name for it. This statement is generated to inform the user of the tags IRACQ has created and the selectional restrictions the user specified for the constituent represented by each tag. Here, the tag ARTICLE has been created for the object slot of the phrase, and IRACQ knows it must be something in the BIBLIOGRAPHY class.

Now IRACQ knows all the constituent slots and the selectional restrictions for them, it needs to know the semantic relations among the constituents. How should the system construct the meaning of a WRITE-ACTION phrase, given the meanings of a PERSON subject and BIBLIOGRAPHY object? The user is asked to use the tags to specify how to

```
Enter predicates, just 'OK' when done.
   Enter a predicate: (AUTHOR JONES ARTICLE)
   Enter a predicate: (IN-JOURNAL ARTICLE NEWSWEEK)
   Enter a predicate: OK
```

Predicates represent relations between the semantic classes in the domain. For now, we assume the user knows something about what predicates are understood by the system. IRACQ tags are used to specify how to instantiate the predicates when the semantic knowledge is used.

5.3 Domain Dependent Semantic Knowledge?

Consider a natural language processor which is a component of some larger system. Given an English sentence, the goal of the language processor is to produce a formal representation of its meaning, an expression in some target language. This representation is then further processed by the larger system to produce the appropriate response. In the IRUS system, for example, a meaning representation is compiled into a database query.

Much of the knowledge needed by an English processor is language specific, rather than domain specific. This includes both syntactic and semantic information. The grammar of a language and the syntactic features of words are (almost) domain independent. The semantics of quantifiers, determiners, and other "closed class" words tend to be domain independent. In addition, a language has semantic facts about how the different syntactic categories behave.

What has to be acquired for each new domain? There are three kinds of domain dependent semantic knowledge.

- domain model. What kinds of things will be talked about in the domain? What generalizations of the classification will be useful? What are the relations

between the classes? For instance, in the example of the last section, things were classified into PERSONNEL, BIBLIOGRAPHY, and JOURNAL. PERSONNEL is generalized by the classes PERSON and AGENT, and the AUTHOR relation holds between the PERSON and BIBLIOGRAPHY classes. The discussion of domain model acquisition is deferred to Section 5.7.

- interpretability. Which of the possible syntactic constructions make sense? What are the selectional restrictions, the kind of thing which should fill each syntactic slot in a phrase? The processor must be able to recognize that, in a particular domain, a phrase of the form "PERSON-in-DEPARTMENT" is meaningful, but not "PERSON-in-BOOK". The example in the previous section established the NP-V-NP-PP syntax as meaningful when the verb is "write", "author", or "publish" and the first NP refers to members of the PERSON class

- predication How should the constituents be combined to form the interpretation of the phrase?

Each class in the domain model is a semantic class. For each class, the processor must be able to recognize (and produce meaning representations of) the phrases that refer to entities in that class and relate them to other classes. To do this, the processor will use a "phrase description", generated by the domain dependent semantic acquisition One way to think of a phrase description is as an abstraction of all the ways of talking about a semantic class. Each time a phrase refers to something in the class, then, it will be a specific instance of our abstraction. A phrase description associates relations between the semantic classes with constituent syntax and selectional restriction configurations.

The phrase description produced for the WRITE-ACTION class in the first section includes

- **the head words.** "write", "author", and "publish".

- **the PEOPLE to BIBLIOGRAPHY relation.** Expressed by a PEOPLE subject and BIBLIOGRAPHY object in a WRITE-ACTION phrase. This relation is (AUTHOR subject object).

- **the BIBLIOGRAPHY to JOURNAL relation.** (IN-JOURNAL object pobj[9]).

A phrase description only needs to identify the most basic syntactic form which expresses the restrictions and relations being defined. Complex constructions such as relative clauses and questions are derived from the simpler ones by the language processor using rules of the grammar.

5.4 System Descriptions

Both IRUS and PSI-NIKL use the RUS [3, 6] system, whose main components are a large ATN grammar, a nearly deterministic parser, and a dictionary and lexical acquisition component. Any semantic interpreter can be interfaced to RUS because RUS cleanly separates its syntactic analysis from the interpreter. This design facilitates RUS's use in many applications. The interpreter may use any convenient semantic representation and target language.

5.4.1 IRUS

IRUS is a system for database retrieval using English queries. Phrase descriptions in IRUS are represented as pattern action rules, called Interpretation Rules or IRules. The target language of the natural language processor is called MRL [11], for Meaning Representation Language, which gets compiled into a database query. A brief description of the domain model, MRL, and the IRules will give the reader a better understanding of IRACQ's goals.

---

[9] object of the preposition

5.4.1.1 IRUS Domain Model

IRUS's domain model uses a variety of representations to express semantic properties. Hence, the word groups used in a phrase description may be expressed in different ways, depending on the word's syntactic category. Adjectives and head words use semantic properties in their dictionary entries (e.g., COLOR for "red" and "blue", and WRITE-ACTION for "write", "author", and "publish"), prepositions use the ONE-OF test shown below in the example IRule (e.g., "in" and "for" as prepositions for IN-JOURNAL phrases). The subsumption relation between classes has a separate representation. Furthermore, other relations between classes (e.g., the IN-JOURNAL relation between the BIBLIOGRAPHY and JOURNAL classes) are not explicitly represented.

5.4.1.2 IRUS Phrase Description

IRules characterize the set of possible constituents in a phrase and the effect each one has on the interpretation of that phrase.

```
(CLAUSE HEAD *
        SUBJECT (SUPERC PERSON)
        OBJECT  (SUPERC BIBLIOGRAPHY)
        PP ((PP HEAD (ONE-OF IN FOR)
                POBJ (SUPERC JOURNAL))))
```

FIG. 2. THE LEFT HAND SIDE OF THE IRULE ACQUIRED IN SECTION 5.2

The left hand side of an IRule, shown in Figure 2, encodes "what is interpretable", specified by a syntactic pattern, e.g., SUBJECT along with a test that can determine whether each constituent is in the right semantic class, e.g., (SUPERC PERSON). These tests express the selectional restrictions for the domain. SUPERC tests that a constituent is subsumed by a semantic class. ONE-OF tests that a word constituent is a member of a given list.

In the example IRule, there is no test for the head constituent because the selectional restriction for a head constituent is actually in the dictionary, not the IRule. All the head words in the phrase description are given a special property in their dictionary entries to indicate their role in the IRule.

```
(BINDQUANTS ((JONES (OPTIONAL SUBJECT))
             (ARTICLE OBJECT)
             (NEWSWEEK (OPTIONAL (PP 1 POBJ)))
            (SEM (QUOTE (AUTHOR JONES ARTICLE)))
            (SEM (QUOTE (IN-JOURNAL ARTICLE NEWSWEEK)))))
```

FIG. 3. THE RIGHT HAND SIDE OF THE EXAMPLE IRULE

The right hand side of an IRule, shown in Figure 3, has three parts: a quantifier function (LIFTQUANTS or BINDQUANTS which will not be discussed in this paper), a list of variable bindings (e.g. (NEWSWEEK (OPTIONAL (PP 1 POBJ)))), and a list of interpretation commands (e.g. (SEM (QUOTE (AUTHOR JONES ARTICLE))))

The list of variable bindings serves several functions. It chooses the constituent pieces from the left hand side which will be needed to build an interpretation and gives each one a name. For instance, the variable NEWSWEEK is bound to the interpretation of the constituent which matched the POBJ of first element in the list of PPs in the left hand side of the IRule. The tags generated by IRACQ in the first section correspond to these names. All of this is domain independent information about the way constituents behave semantically. In addition, the variable binding specifies if a constituent is optional, i.e., does not need to be present for the matrix to have an interpretation.

The final portion of the right hand side, the interpretation commands, are semantic interpreter functions which will build the appropriate MRL when the IRule is invoked. The arguments to these functions express the predication information needed for the domain. The commands will cause predicate instantiations which express the relations between the constituents.

5.4.1.3 IRUS Target Language

The general form for an MRL expression is

(FOR <quant> X | <class> . (p X) . (q X)) where

- <quant> is a quantifier such as SOME, ALL, THE,
- X is the variable of quantification,
- <class> is the domain of quantification, one of the established classes,
- (p X) is a predicate that further restricts the domain of quantification,
- (q X) is the expression being quantified, either a predicate involving one or more quantified variables or an action to take with each member of the domain.

(p X) and (q X) may themselves be quantified expressions. (See [10] for more information on MRL.)

```
(FOR SOME B1 | PERSONNEL : (LASTNAME B1 Jones);
     (FOR A A2 | BIBLIOGRAPHY : T;
          (AUTHOR B1 A2)))
```

FIG. 4 MRL FOR THE EXEMPLAR "JONES WROTE SOME ARTICLES"

The target interpretation in Figure 4 shows the level of understanding needed by IRUS. Paraphrased in English, the interpretation means "Given some bibliographic item

and some member of personnel whose last name is Jones, the AUTHOR relation holds between the bibliographic item and the personnel member." This is compiled into a database query that searches for "Jones" in the author field of a bibliography table.

5.4.2 PSI-NIKL

PSI-NIKL[10] is a system used at BBN in cooperative planning assistance research. A semantic class, a phrase description, and an interpretation are each represented as a NIKL concept. The relation between these three things is also represented in NIKL. |R|INTERP[11] relates a phrase description to its semantic class. The interpretation of a phrase will be a specialization of its phrase description whose |R|INTERP filler is the NIKL concept representing what the phrase refers to, a specialization of the semantic class. The example of these representations in the next three sections will help the reader understand PSI-NIKL acquisition needs.

An important feature of PSI-NIKL is its use of the NIKL classifier in building these representations. Because of this, information is always stored at its most general level of applicability. If an IN/FOR-JOURNAL-PP constituent is present in multiple phrase descriptions, for instance, all will share a concept which represents the structure of this constituent. The various phrase descriptions may or may not individualize the way the constituent is interpreted.

---

[10] PSI-NIKL is the new implementation of PSI-KLONE [4, 5] which uses NIKL, the New Implementation of KLone [8, 9]. PSI-NIKL is not yet fully implemented.

[11] the NIKL role labelled INTERP

5.4.2.1 PSI-NIKL Domain Model

In contrast to IRUS, PSI-NIKL has a straightforward domain model, represented in NIKL. Each semantic class is represented by a concept, and the relations between the semantic classes are represented by roles. For the WRITE-ACTION semantic class, the relevant piece of domain model is shown in Figure 5.

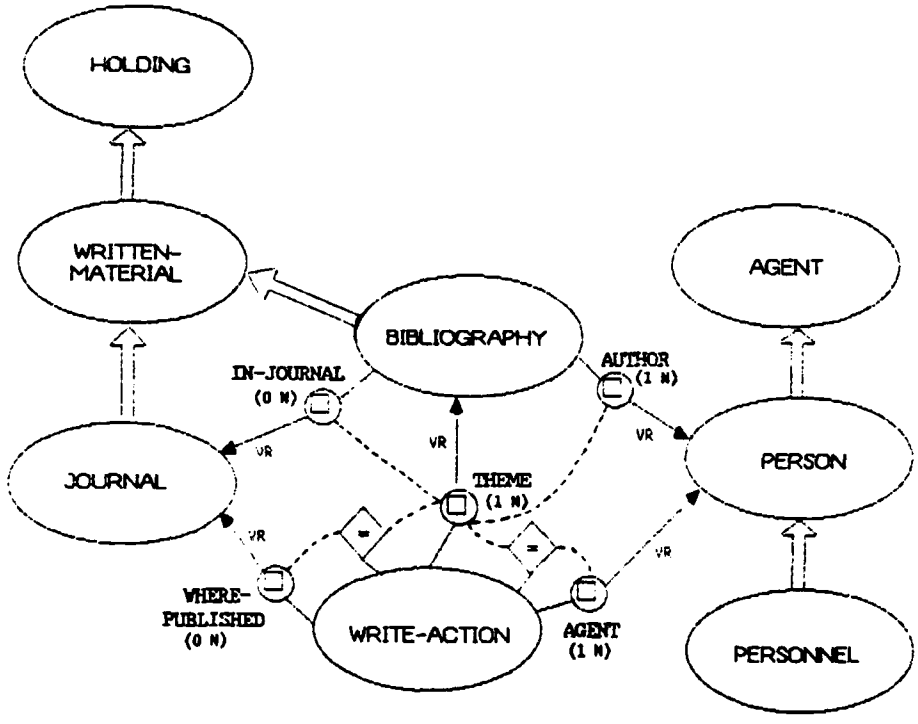


FIG. 5 THE NIKL DOMAIN MODEL FOR A WRITE-ACTION

5.4.2.2 PSI-NIKL Phrase Description

For each semantic class, there is a phrase description concept whose |R|INTERP ValueRestriction is the concept for that class. For each constituent in the phrase description, there is a role whose ValueRestriction (another phrase description) expresses the selectional restriction for that constituent.

In PSI-NIKL, predication is very uniform. For each constituent, there must be a role in the matrix interpretation, |R|TARGET, constrained to be the same as the constituent's |R|INTERP. These are expressed as RoleConstraints, shown in Figure 6.

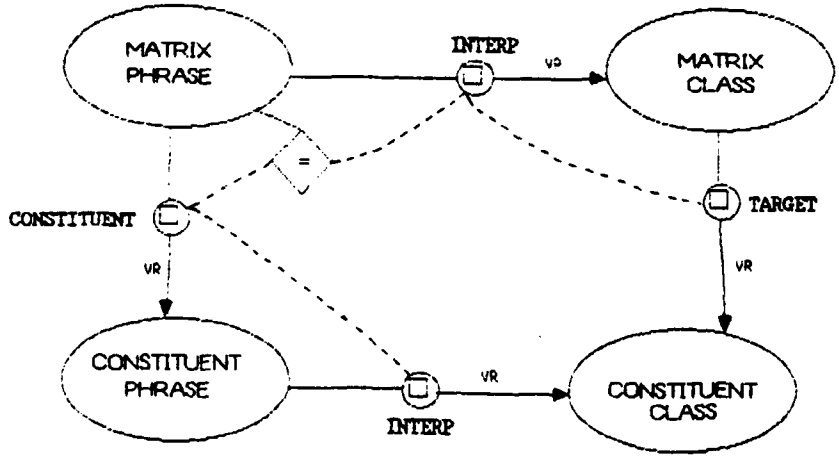
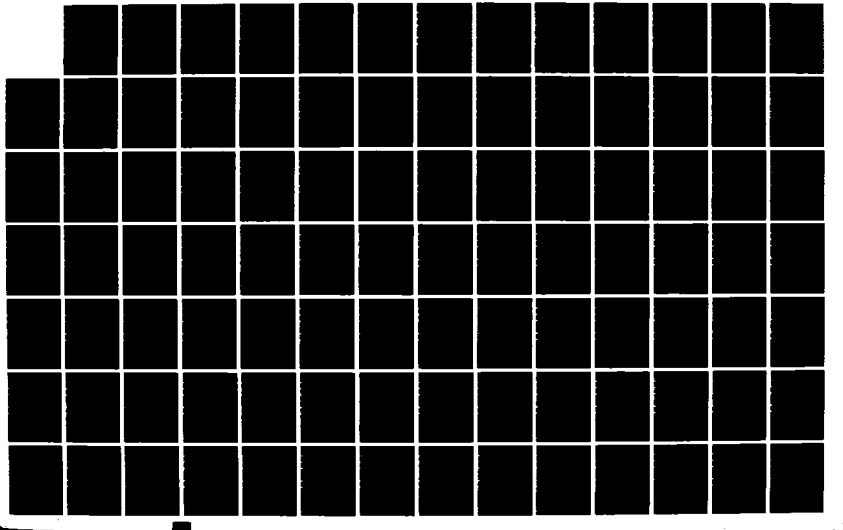


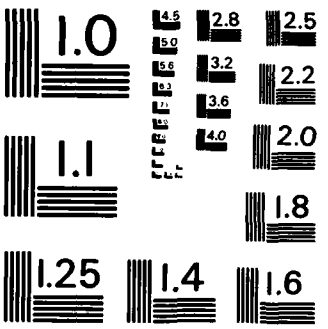
FIG 6 PREDICATION IN NIKL

IRUS does not have this uniformity in how predication is expressed. Recall that the WRITE-ACTION IRule establishes a relation between two constituents, the referent of the WRITE-ACTION phrase is not represented explicitly. For IRUS, clauses usually relate constituents to each other, while NPs relate constituents to the matrix interpretation. Thus, whereas in IRUS a WRITE-ACTION phrase expresses the AUTHOR relation between subject and object, two relations are expessed in PSI-NIKL, the AGENT relation between WRITE-ACTION and the subject's interpretation and the THEME relation between that WRITE-ACTION and the object's interpretation. In PSI-NIKL, the constituents are related to each other by RoleConstraints on the semantic class in the domain model, as shown in Figure 5.

AD-A145 807 RESEARCH IN KNOWLEDGE REPRESENTATION FOR NATURAL LANGUAGE UNDERSTANDING(U) BOLT BERANEK AND NEWMAN INC CAMBRIDGE MA C SIDNER ET AL. SEP 84 BBN-5694 2/3

UNCLASSIFIED N00014-77-C-0378 F/G 5/7 NL





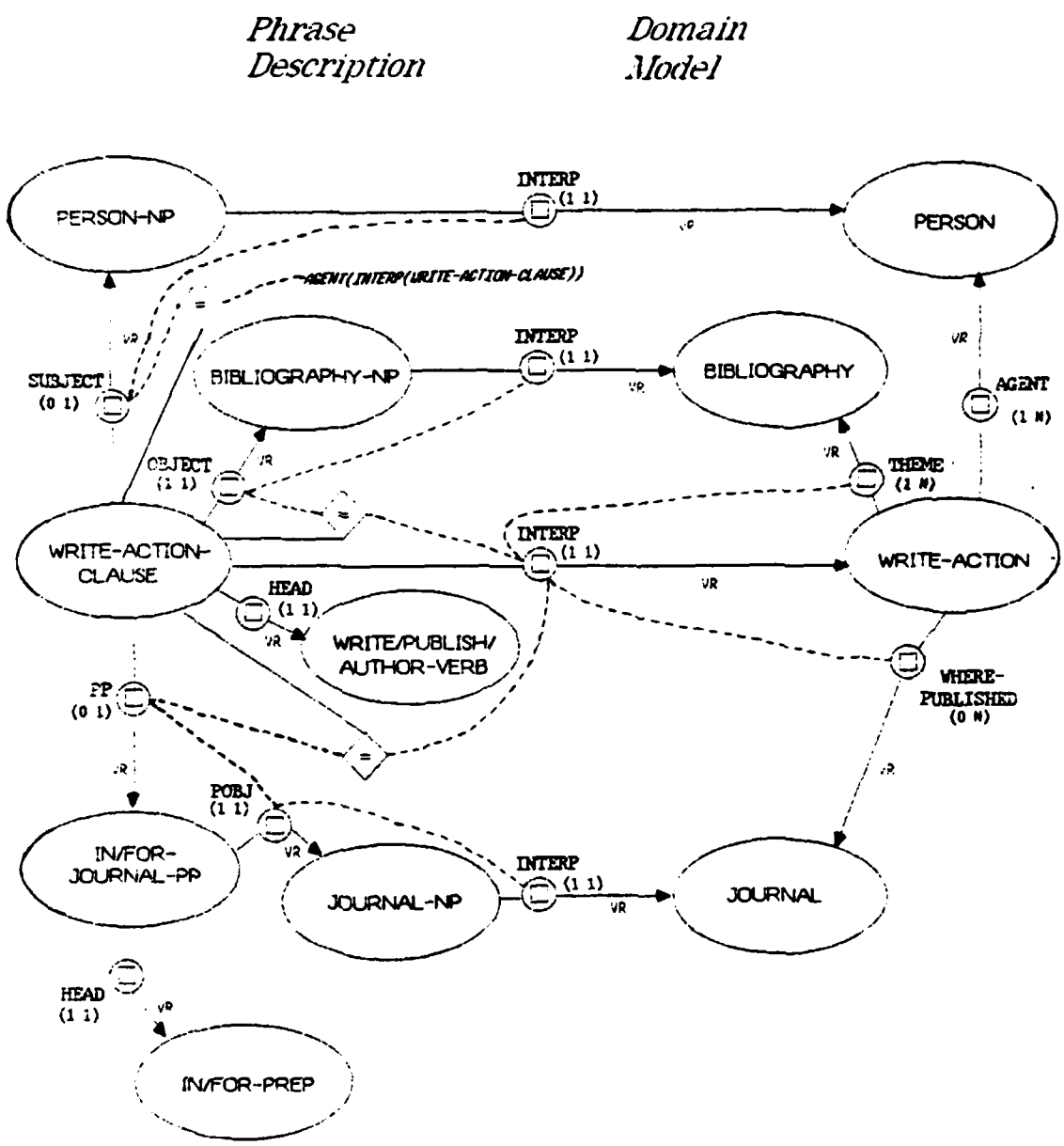


FIG 7. THE WRITE-ACTION PHRASE DESCRIPTION IN NIKL

Notice |C|IN/FOR-JOURNAL-PP in Figure 7. This is a convienent unit which may be

shared by other phrase descriptions. For instance, since the constituent may be either clausal or NP internal, the BIBLIOGRAPHY phrase description will also have this constituent

5.4.2.3 PSI-NIKL Target Language

Recall that phrase descriptions are generalizations (in NIKL, these generalizations are expressed as structured definitions) of the set of phrases which can refer to a semantic class Many subsets can also be defined For example, WRITE-ACTION phrases with PPs, WRITE-ACTION phrases which use the head word "author", etc are subsets of the WRITE-ACTION-CLAUSE in Figure 7 In fact, PSI-NIKL interpretation is the process of defining the subset of phrases, a description concept, which is so specialized that it represents exactly the input phrase The |R|INTERP for this subset, also constructed during interpretation, will represent the subset of the semantic class which can be refered to by that phrase

Figure 8 shows a specialization of the WRITE-ACTION phrase description for all phrases which are exemplified by "Jones wrote some articles" Notice |R|INTERP for this concept represents just those write actions whose agent is a person named Jones and whose theme is some articles written by that person Each constituent will be a specialization of its roles's ValueRestriction on the phrase description Each constituent's |R|INTERP filler is a specialization of its target role's ValueRestriction in the domain model

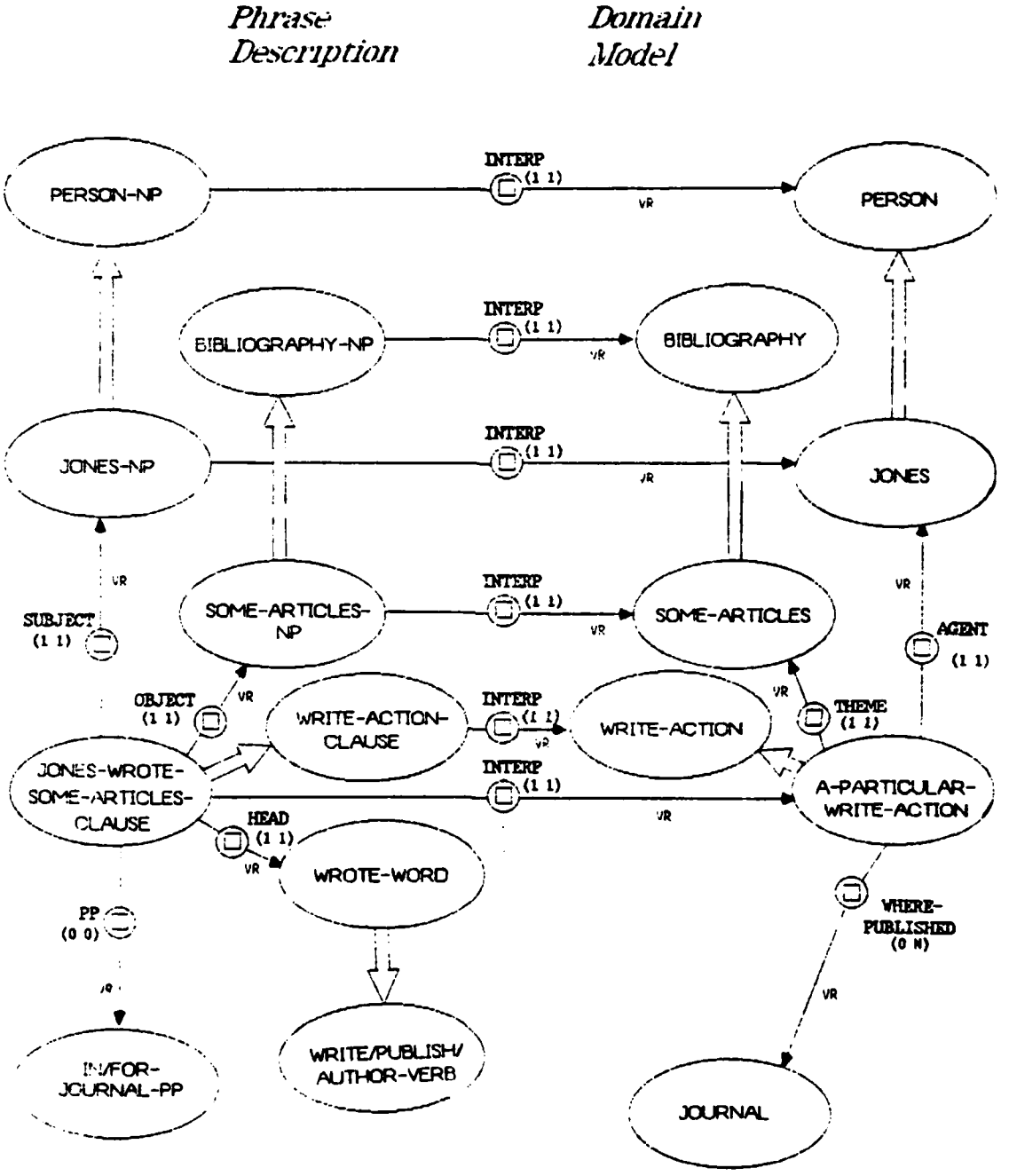


FIG 8 THE PSI-NIKL INTERPRETATION OF "JONES WROTE SOME ARTICLES"

5.5 IRACQ, Semantic Acquisition for IRUS

The IRACQ approach is example driven rather than model driven An important feature of the system is that it isolates just the part of the IRule that constitutes the domain dependent semantic knowledge All of the detail of the IRule representation, much of which is system specific or language specific, is handled by IRACQ Furthermore, the exemplar will constrain the possibilities when acquiring the selectional restrictions, so the user does not need to know semantic class names

When the user enters an exemplar, IRACQ uses the full power of IRUS to interpret it Actually, IRUS does not have the necessary IRules for interpreting the exemplar, so IRACQ forces it to use a vacuous "acquisition IRule" The resulting "interpretation" consists of the syntactic structure of the exemplar and the interpretation of all its constituents This allows IRACQ to aquire the kinds of semantic knowledge outlined in Section 5 3 in a natural way

- interpretability Assuming the user has entered a meaningful phrase, the syntax of the exemplar "points at" constituents which will be interpretable The classes of an exemplar constituent constrain the possible selectional restrictions The user must say which of the classes that the constituent belongs to is the appropriate one to use as a selectional restriction
- predication The tags created by IRACQ correspond to the names in the variable bindings list The user is shown the list of tags, and asked to enter predicates with the tags to indicate how to instantiate them The system handles the domain independent details of what interpretation commands should be written into the IRule

The technique of forcing the IRUS language processor to do partial interpretations requires IRACQ to predict what syntactic configurations might be in the exemplar The acquisition IRule must provide a slot for all possible constituents IRACQ allows for two phrase types, clause and noun phrase, with the following slots

- **CLAUSE**. head (main verb), subject, object, indirect object, PPs, and certain adverbials (e.g. time)
- **NOUN PHRASE**. head (main noun), adjectives, noun-noun modifiers, and PPs

Of course, RUS can parse many other constructions. These have been adequate for the IRUS test domain. IRACQ could easily be extended to handle other configurations.

5.6 Semantic Acquisition for PSI-NIKL

Much of the approach used in IRACQ will transfer easily to semantic acquisition for PSI-NIKL by using the following mapping from IRule representation to NIKL phrase description:

IRule	NIKL
constituent syntactic pattern	a role on the phrase description concept
semantic test for a constituent pattern	the ValueRestriction of the constituent role
the quantifier function	handled in PSI-NIKL
variable binding	the left RoleChain in Figure 6
optional/required	the constituent role number restriction
interpretation commands	a RoleConstraint for each constituent, one RoleChain will be the binding path, the other the path to the target role

An acquisition concept can easily be constructed, just like the acquisition IRule for IRACQ.

The use of NIKL representation will improve both aspects of semantic acquisition.

- interpretability. The selectional restrictions can be constrained using exemplar constituents in much the same way as in IRACQ. Each word group and constituent will be uniformly represented as a concept, so implementation of this part of acquisition will be simpler and more general. As a result, it will be easier to extend the system to acquire semantic knowledge involving new syntactic constructions.

- predication. Because phrase descriptions are explicitly related to semantic classes in a straightforward way, we can use the exemplar to constrain the possible target roles during this phase of acquisition. Upon acquiring the subject constituent for the WRITE-ACTION phrase description, for example, the system can check the domain model for relations between WRITE-ACTION and PERSON. Furthermore, the system can acquire the predication information at the same time as the interpretability, eliminating the need for tags. This could not be done in IRACQ because IRUS predicates could relate either constituent to constituent or constituent to matrix.

  In short, the system can take much more initiative in predication acquisition. Once the selectional restriction for a constituent has been elicited, the user need only select the appropriate target role from the choices generated using the exemplar.

NIKL also introduces complications, however. Because information is shared whenever possible, many generalizations will be automatically created and installed in the taxonomy. PSI-NIKL needs a scheme for marking exactly which phrase concepts are interpretable and which are uninterpretable. Once developed, the task of marking concepts as they are created will fall to the semantic acquisition component.

On the other hand, sometimes these generalizations may provide convenient shortcuts for the user. The user should not have to respecify the information for (C:IN FOR-JOURNAL-PP once it has been constructed. Exactly which concepts will be useful in this respect and how to present them to the user remains to be investigated. For constituents such as PPs, where the needed information actually concerns its subconstituents (i.e., preposition and pobj), the system can first try constraining the constituent using the classification of the whole exemplar constituent. In the example

acquisition, the user might first be asked if the constituent should be a IN-NEWSWEEK-PP, an IN/FOR-NEWSWEEK-PP, an IN/FOR-JOURNAL-PP, or an IN/WITHIN-WRITTEN-MATERIAL-PP. If none of these were the appropriate constituent description, then the system would ask about other prepositions and classes for the pobj.

5.7 Domain Porting

So far the discussion has assumed an already defined domain model. When porting to a new domain, the first step of semantic acquisition will be to construct a domain model.

If the language processor is being used by a database system, as IRUS is, a new domain model can be constructed automatically be examining the structure of the database. This is the approach used by TEAM [7]. However, because it separates domain model construction from the rest of semantic acquisition, IRACQ methods can be used for systems whose goal is not database access. IRACQ constructs a new domain model by prompting the user for the names of semantic classes and their subsumption relations.

Designing a new domain model in PSI-NIKL is difficult because the desired behavior of the system as a whole is not as clear cut as in IRUS. The CKLONE language for constructing networks[12] is one necessary tool. In addition, it would be useful to have tools with some knowledge of domain models, to help the user to insert the

---

[12] See the article by Stallard, this volume.

RoleConstraints (such as AGENT(WRITE-ACTION) = AUTHOR(THEME(WRITE-ACTION)), shown in Figure 5) and DisjointnessClasses (such as the restriction that nothing is both animate and inanimate[13]) which PSI-NIKL needs.

Once a new domain model is constructed, the next step is to acquire the phrase description for each semantic class in the model. Now a problem arises from the assumption that all constituents of an exemplar phrase are interpretable. Presumably the classes in the domain can be inter-related in many ways, so there is no way to guarantee interpretations for all the constituents in an exemplar.

For example, in the phrase description for the PERSONNEL class, a way to talk about the IN-DEPT relation between the PERSONNEL and DEPARTMENT classes is defined. In a new domain, there would not be an IRule to interpret the DEPARTMENT constituent. Trying to avoid this problem by defining the DEPARTMENT phrase description first will not work, because that IRule must define another way to talk about the IN-DEPT relation (e.g., "the department of Jones").

IRACQ solves this by creating a "stand-in" for each class as it is added to the domain model. In an exemplar, the user may use, instead of a phrase constituent, a stand-in constituent to indicate the semantic class for that slot.

---

[13] DisjointnessClasses are used by PSI-NIKL to reject a constituent for a slot in the matrix. For example, if "Jones" were proposed for |R|OBJECT in |C|WRITE-ACTION-CLAUSE, it would be rejected because an animate thing cannot possibly be interpreted as a bibliographic item.

When all the phrase descriptions have been defined, the language processor has all it needs to generate meaning representations for the new domain. There is another phase of domain porting, however. The larger system using the language processor must be instructed how to process the new representations. For IRUS, IRACQ creates new IRules which, when invoked, may produce MRL which the MRL compiler will not be able to handle

5.8 Further Work

One way to extend IRACQ's capabilities would be to improve its facility to acquire a domain model. The construction of a domain model could be cascaded with the rest of semantic acquisition, eliminating the need for the stand-ins. That is, when a constituent is uninterpretable, IRACQ could use an acquisition IRule to force its vacuous interpretation and then interact with the user to create the semantic class referenced by that constituent. Now that a new semantic class has been placed in the domain model, IRACQ can use the constituent and its vacuous interpretation to begin creating its phrase description. The major difficulty with this scheme is finding a way to distinguish the uninterpretable constituents from incorrect parses of interpretable ones

As discussed in the last section, an important extension to the IRACQ work is a tool for MRL acquisition This tool would allow IRUS to acquire specifications of how to process the MRL generated for the new domain.

Example directed acquisition is a natural, user-friendly approach to semantic acquisition However, it puts the burden of completeness on the user. During the

acquisition dialogue, the user may give many exemplars when a phrase description has many constituents There is no guarantee, though, that all the relevant ways of talking about the semantic class and relating it to other semantic classes will occur to the user during this process To cover this shortcoming, we would like to build an IRule editor which uses the IRACQ approach to extend and change IRules. In the case of PSI-NIKL, the system could take more initiative and examine the domain model for relations which are not yet target roles in the phrase description.

REFERENCES

1. Bates, Madeleine and Bobrow, Robert J. A Transportable Natural Language Interface for Information Retrieval. Proceedings of the 6th Annual International ACM SIGIR Conference, ACM Special Interest Group on Information Retrieval and American Society for Information Science, Washington, D.C., June, 1983.

2. Bates, Madeleine. Accessing a Database with a Transportable Natural Language Interface. Submitted to the First International Workshop on Expert Database Systems.

3. Bobrow, R.J. The RUS System. BBN Report 3878, Bolt Beranek and Newman Inc., 1978.

4. Bobrow, R.J. and Webber, B.L. PSI-KLONE - Parsing and Semantic Interpretation in the BBN Natural Language Understanding System. CSCSI/CSEIO Annual Conference, CSCSI/CSEIO, 1980.

5. Bobrow, R.J. and Webber, B.L. Knowledge Representation for Syntactic/Semantic Processing. Proceedings of The First Annual National Conference on Artificial Intelligence, American Association for Artificial Intelligence, 1980, pp. 316-323.

6. Bobrow, R. and Bates, M. The RUS Parser Control Structure. In *Research in Knowledge Representation for Natural Language Understanding. Annual Report*, Bolt Beranek and Newman Inc., 1982. BBN Report No. 5188.

7. Grosz, B. et.al. TEAM: A Transportable Natural-Language System. No. 263, SRI Artificial Intelligence Center, April, 1982.

8. Moser, M. G. An Overview of NIKL, the New Implementation of KL-ONE. In *Research in Knowledge Representation for Natural Language Understanding - Annual Report, 1 September 1982 - 31 August 1983*. Sidner, C. L., et al., Eds., BBN Laboratories Report No. 5421, 1983, pp. 7-26.

9. Schmolze, J. G., and Israel, D. J. KL-ONE: Semantics and Classification. In *Research in Knowledge Representation for Natural Language Understanding - Annual Report, 1 September 1982 - 31 August 1983*. Sidner, C.L., et al., Eds., BBN Laboratories Report No. 5421, 1983, pp. 27-39.

10. Stallard, David. Data Modeling for Natural Language Access. Submitted to the First International Workshop on Expert Database Systems.

11. Woods, W.A. Semantics and Quantification in Natural Language Question Answering. In *Advances in Computers*, M. Yovits, Ed., Academic Press, 1978, pp. 1-87.

6. SPEAKERS' PLANS AND DISCOURSE

Candace L. Sidner

6.1 Introduction: Discourses and Intentions

A discourse is structured into one or more units, each of which expresses some information about a response that the speaker intends for the hearer to produce. The information expressed in a discourse unit specifies the speaker's intention. The actual response the hearer produces based on the speaker's intention is the intended response. The intended response may be as simple as the hearer believing a proposition or as complex as riding a bicycle for the first time. For these two cases the speaker's intention may be exactly the intended responses. However, the speaker's intention may differ from the hearer's response. The speaker may intend that the hearer do whatever act or acts are needed to produce a state the speaker wants achieved (as in the flowers being on the table). Or the speaker may intend that the hearer perform some general act (such as going to the store) without caring what specific response the hearer uses to perform the general act (walk to the store, drive to the store, etc.). In such cases we want to say that the hearer is responding as intended, but his response is not the same as the speaker's intention.

To determine an intended response, the hearer must undertake a complex reasoning task that includes many subtasks. First, the hearer must determine which utterances in a discourse form a unit, a unit in which information about the speaker's intention is expressed. Second, the hearer must extract the speaker's intention from the information in the unit. Third, the hearer must create a description of what his intended response will be.

In this paper I will report on further progress in understanding the speaker's intentions in discourse and in specifying an intended response. I will assume the basic model reported in [8], and will detail aspects of that model not previously explored. In this paper I will also discuss aspects of discourse that are outside the theory of speaker intention recognition and that provide information critical to this theory and the associated processing model.

A discourse unit will either be atomic, if all the utterances in the unit provide information about one speaker intention, or non-atomic, if the discourse is structured into sub-units, each of which specifies an intention. In the atomic discourses below, the speaker expects a single response from the hearer. Sing a particular song in a particular way.

(1) 1 I want you to sing me a song.
2 It's "Yankee Doodle Dandy" in the key of C.

(2) 1 How about *singing me a song*?
2 "Yankee Doodle Dandy" in the key of C.

(3) 1 I want you to sing me a song.
2 I want you to sing "Yankee Doodle Dandy"
3 a) I want it sung in the key of C;
b) Sing it in the key of C.

In all of these discourses the speaker uses more than one utterance to specify the intended response. Each utterance specifies some part of the response--describing the actual act to be performed or expressing a belief the hearer must adopt to carry out the act. The intended response for the whole unit is the hearer's singing Yankee Doodle in C.

By contrast, in (4), the speaker specifies two intended responses, namely, for the hearer to sing Yankee Doodle in C but to first tell the speaker whether singing Yankee Doodle in C is an ability of the hearer. Thus (4) is not an atomic discourse.

(4) 1 How about singing me a song?
2 Yankee Doodle.
3 Can you do it in the key of C?

In D1, the speaker has produced a non-atomic discourse comprised of several atomic units whose intended response is that the hearer adopt certain beliefs. The intended response for the whole discourse is for the hearer to use those beliefs to conclude that M is a dastardly character. The reader must conclude on the basis of the second paragraph that M is cowardly, on the basis the third paragraph that M is cruel and on the basis of the final paragraph that M is power hungry and ruthless, and that those characteristics are evidence for M's being a dastardly sort.

Discourse 1

M stood by the window while planning his next day's pursuits. You could not discern in his brooding face what others said about him. There were stories, never ending it seemed.

M's one-time friend J told the occasion on which their friendship ended. M had promised to back-up J in the deal with Marley Enterprises but when the final meeting came, M begged off on the grounds that his other financial interests put him in ethical conflict with J's plan. Privately, J concluded that M was afraid to back a risky business venture before such giant businessmen as the board of Marley Enterprises.

M's sister, L, would tell of M's childhood pleasure in kicking his dog, dumping his cat in water and pulling feathers off of captured live birds. L would always shudder.

And then there was K, M's ex-wife. Among her remembrances were all the occasions M plotted carefully his next promotion in the office. Each promotion was obtained by producing apparent evidence of a peer's incompetence at his job. Just such a plot was what M was ready to develop now.

A discourse that is rich and complex will be comprised of several units The overall intention communicated by the whole discourse will be conveyed by the intentions in each unit and its relation to the other intentions. By way of illustration, consider the story just mentioned. The overall intention is to get the hearer to conclude something about M. Each unit conveys some aspects of M's behavior, and at the last utterance, the hearer is expected to link the units together and draw the overall intended conclusion about M's character.

These examples of discourse raise many questions.

- How does the hearer infer the speaker's intention within a unit?
- Since an atomic unit can be comprised of more than one utterance, how does the hearer determine what each utterance contributes to the intention of that unit?
- How does the hearer find the beginning of a unit, and the end of one?
- If there are several units, what links them explicitly or implicitly together in the discourse and in the mind of the hearer as understanding occurs?

In this paper, I wish to concentrate on the first two questions, by elaborating the theory and model of speaker intention recognition As I will show, we can reach some tentative conclusions about the third and fourth questions and provide some specific directions for further research.

6.2 The Model for Speaker Intention Recognition

The overall view of understanding discourse I will describe in more detail is as follows. The speaker uses utterances to express his intentions. Some utterances specify one intention while other utterances specify a different one. The hearer must

determine which utterances specify a particular intention and judge where those utterances end in the discourse. Fortunately for the hearer, the speaker organizes utterances into a group (a discourse unit) around each intention rather than dropping utterances about each one throughout discourse. As we shall see, the speaker also aids the hearer's understanding with additional markers for the beginning and end of units. These markers make it possible for the hearer to segment the discourse into units, and to determine the relations among units, and to distinguish the relations among the intentions that are conveyed in those units.

For the moment, let's assume the hearer has determined where a unit begins, and let's turn our attention to how the hearer determines the speaker's intention and a description of his own intended response.

To respond to a discourse, the hearer's initial task is to deduce a set of "speaker wants" from an utterance by using knowledge of the conventional Gricean rules of English (see [8] for a description of these). These wants are related to the speaker's intention because the speaker is intending to tell the hearer what he wants. By speaking intentionally, the hearer can assume that whatever wants are expressed in an utterance are intentions of the speaker. Given some such set of wants, we may ask how the hearer proceeds to determine an intended response to that utterance and possibly to succeeding utterances. My previous work suggests an approach based on Grice's work that the hearer asks him/herself three questions about those wants gleaned from an utterance.

1. Why does the speaker want that? (plan recognition)
2. What capacities do I have, that the speaker is aware of, that I can use to accomplish what the speaker wants? (response recognition)

3 What must I do to use those capacities for the speaker's wants, if I in fact choose to carry out the speaker's wants. (response planning)

In the first phase of speaker intention recognition, the plan recognition phase, the hearer's answer to question (1) will expand the initial set of speaker wants to a set that includes some of the speaker's plans. The speaker has a plan of action that motivated using a discourse to involve the hearer in a response. By recognizing some aspects of the speaker's plans, the hearer will be better prepared to determine an intended response.

Thus if a speaker tells a hearer, "I want to edit a paper," the hearer can conclude initially that the speaker wants to edit some paper[14]. If the hearer knows certain things about the speaker, question (1) may lead him to conclude that the speaker wants to expand a paper he has written in light of comments on the first draft. Or the hearer may know only that the speaker often wants such things things and not know more.

In response recognition, the second phase of speaker intention recognition, the hearer will take note of capacities that are directly mentioned in the utterance as in discourses 1-3, or capacities that will enable the acts that the speaker wants, (as in turning on the lights when the speaker wants to see something) or capacities that result in the condition the speaker wants (as in "I want the vase on the table.") The hearer's answer to (2) may be influenced by the set of speaker plans derived from

[14]Whether the speaker expects the hearer to have any belief about which paper is a topic explored in [9], and will be deferred in this paper.

answering question (1). To return to the edit example, if the hearer knows which paper is to be edited and that the speaker wants to edit by computer, the hearer can plan to respond based on the information.

Once the hearer has determined a response to an utterance, he must decide how to go about producing that response. This is response planning. Question (3) contains a qualification about choosing to act on the speaker's wants since the hearer always has an option not to do what the speaker wants. The hearer must answer question (3) because use of one's capacities may require some planning before the act can occur. Response planning will not be considered further in this paper.

6.3 Speaker Plan Parsing

Now let us turn to a more detailed model of how the hearer can answer question (1), namely how the hearer determines what other wants, expressed as plans, the speaker has that the hearer is expected to draw inferences about.

We will assume that in any situation the hearer has a set of plans that constitute the potential plans a speaker might carry out. This set of potential plans is shared between speaker and hearer, that is, the speaker is aware of the hearer's knowing about them. The hearer must determine which plans from the potential plan set the speaker is actually using at the time of the utterance. The hearer must infer only those plans that the speaker wants the hearer to know about. The speaker intends for the hearer to know about certain plans because the hearer can use information in them to produce his response. The speaker could tell the hearer which plans are underway, but it is easier not to have to express such information if the hearer can infer it.

The speaker may have other plans that the hearer is unaware of, these plans are relevant to discourse only when a speaker uses a discourse to make them shared between speaker and hearer The hearer may also be aware of speaker plans that the speaker does not know the hearer knows about The hearer may choose to structure his response on the basis of these plans as well, but they do not play a role in the intended response recognition process The speaker does not expect the hearer to rely on these unshared plans in recognizing the intended response simply because the speaker does not know the hearer knows about those plans

To infer the actual speaker plans, called the intended recognition plans, we can construct a plan parsing machine, which takes a representation of potential plans of the speaker and a description of an utterance and determines which plan(s) the speaker is carrying out Notice that such a parser will not be able to determine all the speaker s plans, but just those the speaker wants the hearer to know about The total set of speaker plans may be, and in most cases is, larger than the intended recognition plans, but the hearer cannot necessarily parse the total set because necessary information may be missing To draw an analogy to sentence parsing, a parse tree cannot be constructed if prepositions are missing from prepositional phrases or if the main verb is deleted However certain information may be missing from a sentence while still permitting parsing to occur, but only when that information is recoverable from elsewhere Likewise the plan parser must have information either in the utterance or from some additional inferences that the speaker thinks the hearer can draw in order to find the set of intended recognition plans

A critical assumption in parsing intended recognition plans is that the speaker

has the burden of telling the hearer, in some way, what plans are underway. The speaker must give the hearer, or make sure the hearer has available, all the information to parse the plans the hearer is supposed to recognize. As we shall see, this assumption about intended recognition constrains the search of potential plans.

To represent a plan, I will use a structure such as the one in Figure 9. The plan has a heading that consists of the name of the plan (e.g., edit) and typed variables for all the arguments to the plan. <Agent> specifies the one who edits, <paper> specifies an object of type paper to be edited and <device> specifies the instrument for editing, and <device copy> specifies the machine version of <paper>. A plan may either have no steps (atomic and executable directly), or it may have one or more steps which are in turn names of plans. All plans contain pre and post conditions for each step as well as (possible) pre and post conditions for the whole plan

FIG. 9. THE PLAN TO EDIT A HARD COPY PAPER ON-LINE

```
Edit<agent><paper><device><device copy>
step1 Find <agent><paper>
step2. Locate <agent> <device copy> on <device>
step3. Execute <agent> Emacs <device> <device copy>
Plan preconditions: Exist<paper>, Exist<device copy>
Plan Postcondition: Identical-contents <paper> <device copy>
```

A plan parser uses the acts (called intended acts) that are associated with the speaker's wants in an utterance. Either the speaker want describes an act (by the speaker or hearer) directly, or describes some effect of an act (e.g., I want the flowers in the pot). When an act is directly described, the plan parser can use it to start searching through the set of potential plans. If an act is not directly stated,

but an effect is instead, the parser will first determine the act generalization (an abstraction of the set of particular acts that have the stated effect).

Using either an act generalization or the directly stated act, the parser begins finding plans which contain a step that is the uninstantiated form of the intended act. Given some act1, the parser is looking for plans, say P,Q and R, in which act1 can be instantiated as the first step. Any arguments to the plan that are used in that step are bound to the values of act1. If only one such plan, say P, is found, the parser may use that plan, to find other plans in which P is the first step. If, however, the parser finds several possible plans, it ends its parsing until more evidence can be found. That evidence is either available in succeeding utterances or in additional wants that can be inferred from the present utterance

Why must the parser cease its parsing at this point? Why not proceed with all possible plans in parallel and keep searching for higher plans in the set of plans? Such a process is computationally intractable, there is nothing to delimit the search of plans. To delimit the plans the hearer could make some guesses, say based on his/her knowledge of the domain, but the resulting plan set can no longer be reliably used for producing the eventual intended response, because the plan set contains more than what the speaker expected the hearer to know about. This more general parser might be useful for the hearer to invoke in order to decide some private concern (see [4] or [3] for examples of parsers that function in keyhole recognition[15]).

[15]Keyhole recognition, a term coined by Phil Cohen, describes those circumstances where a person can be thought of as watching a speaker through a keyhole and trying to discern the speaker's plans from his verbal and non-verbal acts. The speaker is not trying to tell the watcher what he's doing.

However, the speaker cannot build a discourse on the assumption that somehow the hearer will guess at the information that is needed for the hearer to make an intended response. The intended recognition plan assumption cannot be overlooked because it greatly reduces the search space for the parser. As we shall see, the speaker's marking discourse unit boundaries will reduce the search even further.

The above description of parsing seems to suggest a bottom-up implementation strategy. However, an implementation must also make use of utterances that simply point to a place in the potential plan hierarchy, with the assumption that succeeding utterances will fill in the plan below that point (as in a discourse where the speaker initially says "I want to draw a new screen display," and then goes about giving a set of commands to do that). To draw an analogy to sentence parsing, such utterances are akin to the speaker being able to say "noun phrase--the big dog".

At some point in parsing, enough evidence can be found to indicate that only P and not Q and R is the speaker's plan. From then and until the plan is fully specified, P may be used both to produce expectations and details the speaker has not specified.[16]

Of course, it is possible that the speaker has two or more plans that s/he is

---

[16] I have built one such parser and am now constructing a second. The new one parses plans represented in the language NIKL [10].

carrying out and attempting to communicate about in an utterance[17] Thus, suppose act1 is the first step of P and Q, both of which are intended recognition plans for the speaker If the speaker intends for the hearer to use the knowledge of P and Q in constructing a response, then the speaker must tell the hearer one way or another that both P and Q are viable. Without this information, the hearer cannot distinguish infering both P and Q from the situation where either P or Q is the plan but the speaker has not indicated which yet

A critical component of parsing intended recognition plans is that the speaker has the burden of telling the hearer, in some way, what plans are underway. The speaker must give the hearer, or make sure the hearer has available, all the information to parse the plans the hearer is supposed to recognize.

The real demand of providing information about all the plans comes when the speaker is making use of more than one plan. This can occur in many ways two plans can be interleaved, a plan may be interrupted temporarily, a plan may be debugged with another plan and plans may be followed by other plans

6.4 A Digression: Discourse Markers and Plans

When two or more plans occur in a discourse, the hearer may not be able to (in fact in general cannot) tell from the intended act associated with an utterance

---

[17]Cases where the speaker has several plans, but intends that only one is to be communicated as an intended recognition plan are not relevant here because the parser is considering only intended recognition plans.

whether it continues the current plan, or starts a new plan that may or may not be related to the current plan. To illustrate this problem, we will digress briefly and consider another aspect of discourse, i.e., those words, phrases and other syntactic discourse markers that signal what relation the intended act bears to other acts and plans already recognized

In the discourse below the speaker's overall plan is to add a new definition to some knowledge base To carry out that plan, the speaker has to check some information the hearer (a computer system) has, and so he tells the hearer to provide that information (D2-2)

Discourse 2

User 1 I want to add a new definition to the knowledge base.
2. First, give me a list of all the users and systems

S 3. Do you want to know which systems each user uses?

User 4. Yes, but just list one if there are several.

S 5. (system prints information out)

User 6. OK, now I want to define a system user as a person who uses any of these systems.
7. Then assert that all the people above are system users.

He notes the relation of the overall plan of adding a definition to the subplan of getting a list of users and systems with two discourse shift markers, a clue word "first" (cf [7]) and a shift of mood from declarative to imperative (cf [6]). The act of getting a list in D2-2 is not to be taken as the initial step of adding a definition. Rather, getting a list is a distinct intended response in a separate plan that is to be accomplished before the main item of business is undertaken. In effect, the user has

announced one plan and put it aside temporarily for another plan, one of gathering information The hearer need not determine the precise relation between the two plans because the speaker has told the hearer what is relevant. get the list first

The list getting request spurs a discourse clarification exchange(D2-3-5). When the intended response is produced by the hearer, the speaker marks the conclusion of the information getting with "OK", and marks the return to adding definitions with "now" and a declarative definition (D2-6) The speaker does not have to tell the hearer that the definition is to be added to the knowledge base because the hearer has already inferred that from the adding a definition plan in D2-1. The definition itself is enough to make clear that the adding a definition plan is now in effect and that the definition is to be added

The clue word "then" plus mood shift in D2-7 indicates yet another intended act, one of asserting who the system users are for the moment. By itself, the act of asserting that a group of people are system users may or may not be part of the adding a definition plan. The speaker clearly indicates that the naming is a distinct intended act that follows the other steps in adding a definition Whether it is the third step of the adding a definition plan or a distinct plan depends on whether the hearer and the speaker already assume that the add-definition plan has two or three steps The discourse markers make clear that the utterance names a new intended act and is not simply further information about the definition of system user.

When are discourse markers required in a discourse? Reichman, who first recognized that clue words mark boundaries in a discourse, never resolves this

question. Cohen [5]illustrates that in arguments clue words make the processing of claim/evidence relations efficient and possible in real-time. They also make it possible for the speaker to change the order that utterances conveying evidence appear in. Polanyi and Scha classify a large number of discourse markers (including particles (clue words), tense shift, meta comments, repetitions, changes in speech act, use of vocatives, renominalizations of pronouns and paralinguistic devices).

Discourse markers are necessary to indicate

1. Whether the utterance should be taken to express a new intended act rather than further specify an act already described, or
2. Whether the intended act is the first step of a new plan rather than the continuation of a plan already in progress

To illustrate these claims, reconsider discourse D2 and drop the clue words "first," "ok, now" and "then". It is no longer possible to tell (1) that giving a list of users and systems is not part of adding a new definition, (2) if wanting to define a system user is a comment on the system's printing response or a new act, and (3) that asserting certain people as system users is not part of how a system user is defined.

The discourse markers are the speaker's way of telling the hearer that the next utterance must be understood as conveying a new intended act or the start up of a new plan. Just which role the utterance plays depends on the plan currently in progress as well as the set of potential speaker plans that the hearer has available to him.

Only one plan-plan relation seems not to require a discourse marker. When an

utterance is used to convey an intended act that is part of a plan that debugs the current plan step, no marker is necessary. Thus, in D2, suppose that after the system printout, the speaker had said "I can't read that printout because the font is too small. Divide the list onto two screens." The speaker is now indicating that he is debugging the system's way of giving a list of users and systems. No markers are used, and yet the comment, its intentions, and its role in the discourse are clear.

A parser can easily recognize this case because of certain conventions in turn taking. Whenever a speaker initiates a plan in which a hearer is to perform some number of the plan's steps, the speaker and hearer are aware that the hearer's response will either be approved or disapproved. Declarative utterances, particularly those stated in the negative can indicate disapproval. At the same time, these utterances can function to describe the condition that the speaker wants to debug. The coupling of a debugging plan with the current plan step can be triggered with just the approval/disapproval expectation. Notice that an utterance that seeks to debug some step other than the current one cannot go without a discourse marker, since no expectation can drive the recognition.

- If indeed discourse markers are necessary to recognize the relations between the intended acts and the overall plans of the speaker that motivated the discourse, then the plan parser must either itself contain a recognizer for these markers or receive information about them from some other process. The latter choice seems preferable to me. A discourse marker parser cascaded (in the style of the RUS-PSI-KLONE cascade [2]) with a plan parser would provide the necessary information.

The plan parser would parse plans while the discourse marker parser would note the structure of the discourse as signaled by the markers. Each time the discourse marker parser (hereafter the dm parser) detected a set of discourse markers, it could use them together with a model of the plan structure (provided from the plan parser) to determine whether a plan is being marked as ended with another plan getting underway, a plan is being interrupted, or a plan is interleaved with another plan or whether no marks have occurred. The plan parser can use these comments to decide in detail how to view the intended act. Thus, if the dm parser interprets an "OK, now" as a plan completion plus a plan startup, the plan parser can decide if the act following "now" is totally new or the next step in a plan mentioned earlier and interrupted (as in D2).

This view of discourse markers and the intended acts associated with utterances transforms the questions I asked previously about the beginning and ends of discourse units to questions about the recognition of the intentions and intended responses that underlie those units. The acts and plans behind utterances are the semantic structures of discourse, while discourse markers are the syntactic, connective tissue.

6.5 Plan Parsing continued

The plan parser described so far produces a set of plans that serve as the semantics for the units of the discourse. Plans flesh out the intent, communicative and non-communicative, of the speaker and partially specify the purposive act the speaker wants the hearer to perform. The specification is partial because the capacities the hearer is to use may or may not be specified in the speaker's plan. They may need to be inferred from other knowledge mutually shared between speaker and hearer.

Once a set of speaker plans has been parsed, the hearer can use them, and the originally inferred speaker wants to answer question (2) of the intended response recognition process. The full plan may provide parameters the speaker has not stated in the utterance as well as a way to interpret ambiguous phrases in the utterance. If what the speaker wants describes an intended act by a hearer, the hearer must ask him/herself whether s/he has a capacity to do the intended act. That capacity is then the basis for planning an intended response. If the intended act is a generalized act inferred from an effect the speaker described, or from a general act the speaker stated (e.g., going to the store), the hearer must choose a specific act that is one the speaker believes the hearer has, or that failing, is one the hearer can do even if the speaker has no particular knowledge of it.

Thus far we have spoken of a plan parser and a theory of intended response recognition by considering the processing of a single utterance to specify an intended response. However, as the discourses about singing Yankee Doodle indicated, it may take several utterances to fully specify the speaker's intention. The hearer must

- determine the scope of the discourse unit,
- decide what intended acts are conveyed in each utterance,
- and decide how these specify the speaker's intention.

If discourse markers play the role I have claimed, then within an atomic unit, the processing of the utterances must provide the plan parser with the proper place to hook each new utterance's intended act into the plan tree recognized so far.

Return to the simple example.

(1) 1 I want you to sing me a song.
2 It's "Yankee Doodle" in the key of C.

If after (1)-1, the plan parser recognizes that the speaker wants the hearer to sing a song as part of the more general plan of being entertained, then when (1)-2 is recognized, the plan parser must decide what to do with the fact that the speaker believes something is Yankee Doodle in the key of C. (This is the propositional attitude to be understood from declaratives using the [8] model.[18] A focusing device will determine that "it" refers the song mentioned in D(1)-1. What, then, is the parser to make of a want of a belief about a song?

The song is not just any song, but rather a song the parser knows to be part of the plan for the hearer to sing a song. It is just this connection between the utterances and their intended acts that is necessary. As for the phrase "in the key of C", since this phrase is a qualifier on the act of singing and not a modifier of "Yankee Doodle", it is only by recognizing that (1)-2 is a belief about the intended act in (1)-1 that the phrase can be made sense of at all.

The internal structure of a discourse unit brings me to a final problem that will not be resolved here. The speakers's plans that we have been considering are mostly non-communicative ones, wanting someone to sing a song, to conclude that M is a dastardly fellow, to give a list of names, etc. In order to get someone (namely some hearer) to do any of them, the speaker must communicate. The speaker must create a

---

[18]Note that this utterance has a prepositional phrase attachment ambiguity to be resolved.

sub-plan for uttering a description of his wants to another who will see that the wants are fulfilled. Each want then requires a discourse plan for how to communicate that want, that is, how to structure the utterance, what descriptions to use, and the like[19]. For richly connected sets of wants, the speaker must determine an overall discourse plan for expressing the connections between all the utterances that communicate the speaker's wants.

If the want is easy to convey, an utterance will do. But if not, or if the mental resources needed for generating the utterance cannot be had, several utterances may be needed, and the speaker will need to construct the utterances one by one. Is the discourse plan a plan that the speaker intends the hearer to recognize? Or is it sufficient that the hearer use the connective tissue (conjunctions, tense and referring expressions and the like) of the discourse unit to determine the intended response without recognizing how the speaker decided to construct the individual utterances?

For the parser that I have explored, one that functions in a domain of task oriented dialogues, recognizing all of the speaker's discourse plan is unnecessary. Generally, just a few utterances are needed to convey the speaker's intention for an action by the hearer, and nothing seems to be gained from discerning the speaker's discourse plan down to the level of the structuring of individual utterances. It is necessary that a discourse marker parser recognize the basic unit structure of the discourse. Whether in stories, where humor seems to depend on certain turns of phrases, or in arguments, where persuasiveness may be governed by style, a more detailed discourse plan is intended to be recognized remains for further consideration.

---

[19] See [1] for one such model of the speaker's plans to structure an utterance.

To conclude this paper, I want to illustrate how a particular plan parser and set of plans can be used to discern the speaker's intention in various units of a dialogue, how the plans and the intention expressed in an utterance join to convey the response the speaker intends, and how a discourse marker parser would provide necessary information to the plan parser

6.6 An Example of Intention Recognition and Plan Parsing

The discourse participants are a user and a system. The system has a knowledge base of items that the user needs to check on, add to, view graphically and edit. The user also occasionally needs to enlarge the categories of items by defining new concepts to store in the knowledge base. The system assists the user by providing displays of the knowledge base, describing items in it (in English), adding the items the user requests, and editing items as the user specifies. The user cannot change the knowledge base directly, but rather must request any changes through the system. The system is aware that the user typically wants to do the tasks mentioned above (and the user knows this about the system) For each such user task, the system has a plan, i.e., a data structure, specifying the steps (many of which are plans of some complexity) to complete the plan, including pre and post conditions. The set of all plans, the user potential plans, are the plans it will use to interpret the user's utterances.

Consider the following dialogue between user and system.

Discourse 3

User 1. Show me the general concept for "employee"

System 2 <Display of concept in mid-screen. The concept has associated terms for first name, last name, Social Security number and job ID number.>

User 3 I can't fit a new instance below it.
4 Can you move it up?

System 5 Yes <moves displayed concept up to top 1/3 of screen.>

User 6 OK, now display a new instance concept for employee
7 The first name is "Sam" and the last name is "Jones"
8 The Social Security number is 111-22-3333

After D3-1, the system recognizes, via rules of the conventional use of English, want1, i.e., that the user wants the system to believe that the user wants the system to show the user a knowledge base general concept called employee. This want is the post-condition of a plan by the user to say an imperative utterance with a show-verb with agent <system>, recipient <user> and object a knowledge base item (<kbi>) (see figure 10). This plan and one like it for declaratives are the user's plan for getting the system to show a piece of the knowledge base (as illustrated[20] in figure 10.). Getting a show is the first step in the add-new-instance-by-graphics and edit-concepts plans given in figure 11. The check-out plan has only one step, getting an inform, which can be either the user getting a show or getting the system to describe in English the knowledge base item.

The plan parser searches through these plans starting with want1 as the post-condition of saying an imperative show utterance. The parser recognizes that it has seen an act that is the first step of the plans check-out any general concept, add-

[20]Only some of the pre- and post-conditions are shown here.

```
Get <user>(Inform <system><user><kbi><screenloc>)
 OR - (Get <user>(show <system><user><kbi><screenloc>))
    - (Get <user>(describe <system><user><kbi>))

Get <user>(Show <system><user><kbi><screenloc>)
 OR - (Say <user>(Imperative-utterance show-verb<system>
          <user><kbi><screenloc>))
      post-condition: <system>believes<user>wants
           Show<system><user><kbi>
    - (Say <user>(declarative-want-utterance
           Show-verb <system><user><kbi>))
```

FIG. 10 PLANS TO SAY GET-INFORM AND GET-SHOW

```
Check-out <user><Kbi><system><screenloc>
   (Get<user><system> (Inform <system><user><Kbi><screenloc>))

Add-new-instance-by-graphics <user><system><Kbi><Kbii><screenloc>
   (Get <user> (show <system><user><Kbi>))
   (Put <user><Kbi><screenloc>)
    precondition of add : (Instance-of-concept Kbii Kbi)

Edit <user><Kbi><system><value><name>
   (Get <user> (show <system><user><Kbi>))
   OR - (Get <user> (Edit <system><Kbi><value>))
      - (Get <user> (Edit <system><Kbi><name>))
```

FIG. 11 PLANS TO CHECK-OUT,ADD-INSTANCE AND EDIT

a-new-instance-by-graphics of a general concept and edit a general concept.

Unable to distinguish which of these is underway, the parsing halts while remembering that the edit, add and check-out plans are its three best alternatives for now. Given no further means of deciding why the user has his current wants, it decides just what capacities to bring to bear on the user's wants (i.e., how best to show the concept), and plans its response. Had the system been aware that in fact the user was going to add a new instance of the concept, it might have chosen to place the information on the screen higher up.

Utterance D3-3 provides the system with two user intentions.

- the user wants the system to believe the user wants to fit a new instance below the general concept.
- and user wants the system to believe the user believes the user cannot fit one there

The plan parser must determine what these wants convey about plans. It is also expecting that at least one of these wants expresses approval or disapproval of the system's previous action.

The definition of the add-instance-by-graphics plan says the user must put a new instance of the general concept at some screen location. (The details of the put plan specify either getting the system to show the instance on the screen or the user using a mouse to draw it on.) Wanting to fit a new instance concept below the general concept is a put act[21] so the parser can conclude that in fact the add-instance-by graphics plan is in effect (the other plans are excluded because their next steps are

---

[21]The parser has a small synonymy lexicon, which it uses to tell it that if it is already trying to match "fit" to "put", the match is okay. The lexicon does not tell the parser what matches to try, but rather approves, or disapproves matches the parser suggests.

different). The parser also asks if this add-instance plan is part of some other plan. Here it fails to find any such plan, and so it turns to the other want, about a belief that the user believes the user can't fit a new instance below the general concept "employee".

This want corresponds to the parser's expectation about some comment on the last action. In fact it is a disapproving comment. The comment is the first step in one of the debug plans the system attributes to the user, namely debugging the step of fitting a new instance in. Normally the parser would expect that the step to be debugged is the display step, but by explicitly naming the next step, the user draws attention to that one.

```
Debug <user><step><condition><system><action>
 OR - (Inform <user><system><condition>)
    - (Inform <user><system> (not (can user<step>))
 (Bring-about<user> (do <system><action>))

Bring-about <user><system><action>.
 (Determineif <user><system> (can <system><action>))
 (Get <user><system> (<action><system>))

Determineif <user><system> (can <system><action>)
 (Get <user><system>(Inform <system><user>
    (Can <system><action>)))
 precondition: (not:[<user>believe('can '<system>'<action>)])
```

FIG 12 PLANS TO DEBUG, DETERMINE-IF AND BRING-ABOUT

Illustrated in figure 12, the plan to debug a fitting in an instance has two steps.

each a plan The first step informs the system that the user can't do a step (and thereby cause the system to believe the user believes the user can't do the step of fitting in), while the second step brings about some action by the system (where the action to be done must have the effect of establishing at least one of the preconditions of the fitting an instance act). The preconditions of fitting an instance at a screen location are that the screen location is empty and that there is enough room for the instance to be placed at that location. When the action to be done is chosen, the parser will have to check that that action establishes one of the above preconditions.

Before the parser can accept the debugging plan as a plan that it is to recognize, it must be certain that there is a fitting step in some other plan. To do so, the parser checks on the relation between this debugging plan with other plans it has recognized (i.e., the add-instance-by-graphics). The parser finds the plans related because the next step of add-instance-by-graphics is the step being debugged. Thus the parser is able to determine a relation between two plans when no discourse marker has appeared in the discourse to indicate that there is more than one plan under active consideration. Because the debug plan is a debugging, the parser also concludes that the add-instance-by-graphics plan is suspended for the moment, and the parser stacks the add plan to return to once the debug plan is completed

The parser also tries to see if debugging is a step in some more abstract plan, but it finds none, so it concludes its decision making until additional utterances occur.

The user continues his comments to the system by asking "Can you move it up?" The system takes this question to mean that the user wants to determine if the system can do an action of moving up the concept called employee. People typically ask "Can you" questions either when they must find out if the hearer can do some action or when they already know the hearer can but want to give the hearer the option of not performing the action. When the system does not know if the user is aware of what it can do, it recognizes the "can you" question as the step in a "determineif" plan (illustrated in figure 12). It also recognizes that the determinif plan is the first step of a bring-about plan. A bring-about plan with an action of moving up is indeed the next step of the debug plan if and only if the effects of moving up establish a precondition of the fitting in step. Since the parser concludes that such effects hold, it recognizes the next (and final step) of the debug plan.

Because bring-about has two steps (see figure 12), namely, the user determines if the system can do a move-up and then gets the system to do a move-up, the system has further expectations to act on. But since it knows the user is bringing about a move-up, when it begins to plan its response, it can decide to perform a move-up (as well as answer the question asked) as a kind of helpful behavior. One may want to argue that, in fact, performing a move-up action is intended because the system knows about the user's more general plan to bring about the move-up. In the case of human dialogue participants, I would agree with this claim, but it's still unclear what intentions people wish machines to perceive about uncompleted parts of their plans.

Once the system responds to the user's requests in D3-3 and 4, the user

continues the dialogue with D3-6. Before the utterance is made, the system already knows that the employee concept has concepts associated with it by relations named "first name", "last name" and "Social Security No." When asked to display a new concept, the system is aware that these relations may be specified.

The first part of D3-6, "ok," serves two purposes, to approve of the last response and to indicate the end of a plan. Since in the system I have built, the plan parser is not coordinated with a discourse marker parser, this latter information is not available, and so the parser must choose its next step by just following what's left in its current plan and suspended plans. Since the bring-about plan and debug plan have no more steps, it considers them complete and returns its attention to the add-instance-by-graphics plan, with the current step as putting some instance at some screen location. Since displaying is putting for this situation, the parser recognizes the utterance's intention (which is that the user wants the system to believe that user wants the system to display an instance concept for employee) as putting the instance concept in the location below the general concept. The user did not state the location of the display in D3-6, but assumes nevertheless that it's known because of the plan recognition that has occurred previously. The parser has the proper location information (i.e., below the general concept) because it was understood during the plan parsing following D3-3.

When the plan parser begins to interpret the beliefs[22] the user wants in D3-7

[22]These beliefs are that the user wants the system to believe that the user believes that the first name is "Sam" and the last name "Jones," and that the Social Security number is 111-22-3333.

and 8. it proceeds just as was discussed for the discourses about singing Yankee Doodle. The parser asks itself why the user wants the system to have these beliefs, and asks for the referents for the definite noun phrases. These phrases refer to the relations of the new instance concept in the step of the add-instance plan being continued. Hence, it takes these user intentions (for the system to believe) to be intended because they further specify the put step in the add-new-instance plan. These intentions do not tell the parser about new plans nor about new steps. They describe an existing plan step further.

This last assertion should seem quite obvious to the reader. Making it obvious to a machine depends crucially on the plan parsing process going beyond the conventional intentions of a declarative utterance. Because the plan parser is aware of a plan with a particular step and because it can recognize what user intentions are expressed in an utterance for that plan, it can conclude that the user wants the system to have beliefs about the last name and first name of the instance being displayed and not simply to have some more beliefs. The system is to use those beliefs in the display act. Using those beliefs in creating a display is the response people would say the user intended. Hence the plan parsing process has helped determine the intended response by recognizing the speaker s plans

This scenario illustrates the role of speaker plan recognition in the intended response recognition It shows that by understanding the speaker's plans, the hearer can determine information about how it is intended to respond that is not directly stated (e.g., the location of below employer for the display of a new concept), and how to chunk several utterances that convey info. mation about one intended response (as

in D3-6-8). By understanding the speaker's plans, the hearer can choose an action that is helpful, but not intended, not on linguistic forms such as "Can you" but on the basis of what the speaker is actually doing.

This scenario also demonstrates the utility in recognizing the units of the discourse structure. The hearer needs to know enough about the discourse structure to judge when the speaker has finished saying all the utterances that convey a particular intended response and thereby to find the end of a discourse unit. Discourse markers help the hearer determine the end of a unit. The discourse markers also provide the hearer with some indirect information about the connections between the various intended recognition plans. Further research will be needed to explore a discourse marker passer and to build it in a cascade, or other process connection, to a plan parser.

This paper has explored a theory of speaker intention understanding in discourse. I have illustrated how a hearer can recognize the overall plan behind a speaker's communications in a discourse. I have shown how the discourse may be divided into units interpreted around the speaker's intention. Parsing the speaker's intention as part of a plan requires information that can be provided by discourse markers. They are needed to reduce the processing load in parsing the intentions conveyed in utterances. I have illustrated that the combination of plan parsing, recognition of the information available in discourse markers and recognition of the units that form a discourse all contribute to determining the response the speaker intended.

6.7 Acknowledgements

I have benefited greatly from the comments of my collegues Barbara Grosz, Andy Haas, Bob Ingria, and Peggy Moser on drafts of this paper.

REFERENCES

1. Appelt, D.E. Planning Natural-Language Utterances to Satisfy Multiple Goals. 259. SRI, March, 1982.

2. Bobrow, R. and Bates, M. The RUS Parser Control Structure. In *Research in Knowledge Representation for Natural Language Understanding. Annual Report*. Bolt Beranek and Newman Inc., 1982. BBN Report No 5188.

3. Brotsky, D.C. An Algorithm for Parsing Flow Graphs. TechnicalReport" 704, MIT, Artificial Intelligence Laboratory March, 1984.

4. Carver, N.F., Lesser, V.R., and McCue, D.L. Focusing in Plan Recognition. Proceedings of the National Conference on Artificial Intelligence, The American Association for Artificial Intelligence. Austin, TX. 1984, pp.

5. Cohen, Robin. *A Computational Model for the Analysis of Arguments*. Ph.D. Th., University of Toronto. 1983.

6. Polanyi, L., and Scha, R. J. H. On the Recursive Structure of Discourse. Proceedings of the January 1982 Symposium on Connectedness in Sentence, Text, and Discourse. The Catholic University of Tilsburg, Tilsburg, The Netherlands. 1983. In press.

7. Reichman, R. *Plain-speaking. A theory and grammar of spontaneous discourse*. Ph.D. Th., Department of Computer Science, Harvard University, Cambridge, Massachusetts. 1981.

8. Sidner, C.L. "What the Speaker Means. The Recognition of Speakers Plans in Discourse". *International Journal of Computers and Mathematics. Vol. 9, No. 1* (1983).

9. Sidner, C.L., Bates, M., Bobrow, R., Goodman, B., Haas, A., Ingria, R., Israel, D., McAllester, D., Moser, M., Schmolze, J., Vilain, M. Research in Knowledge Representation for Natural Language Understanding – Annual Report, 1 September 1982 – 31 August 1983. Technical Report 5421, BBN Laboratories, Cambridge, MA. 1983.

10. Vilain, M. B. and McAllester, D. A. Assertions in NIKL. In *Research in Knowlege Representation for Natural Language Understanding. Annual Report. 1 September 1982 to 31 August 1983*, Report No. 5421, Bolt Beranek and Newman Inc., 1983.

7. REPAIRING REFERENCE IDENTIFICATION FAILURES BY RELAXATION

Bradley A. Goodman

7.1 Introduction

Our goal is to build robust natural language processing systems that can detect and recover from miscommunication. This requires a study on how people communicate and how they recover from problems in communication. This paper summarizes the results of a dissertation [14] that investigates the kinds of miscommunication that occur in human communication with a special emphasis on reference problems, i.e., problems a listener has determining whom or what a speaker is talking about. We have written computer programs and algorithms that demonstrate how one could handle such problems in the context of a natural language understanding system. The study of miscommunication is a necessary task within such a context since any computer capable of communicating with humans in natural language must be tolerant of the imprecise, ill-devised or complex utterances that people often use.

Our current research [32, 33] views most dialogues as being cooperative and goal directed, i.e. a speaker and listener work together to achieve a common goal. The interpretation of an utterance involves identifying the underlying plan or goal that the utterance reflects [7, 1, 30]. This plan, however, is rarely, if ever, obvious at the surface sentence level. A central issue in the interpretation of utterances is the transformation of sequences of imprecise, ill-devised or complex utterances into well-specified plans that might be carried out by dialogue participants. Within this context, miscommunication [26] can occur.

We are particularly concerned with cases of miscommunication from the hearer's viewpoint, such as when the hearer is inattentive to, confused about, or misled about the intentions of the speaker. In ordinary exchanges speakers usually make assumptions regarding what their listeners know about a topic of discussion. They will leave out details thought to be superfluous [3, 21]. Since the speaker really does not know exactly what a listener knows about a topic, it is easy to make statements that can be misinterpreted or not understood by the listener because not enough details were presented. Two problems that could be encountered by the listener during interpretation of an utterance include incorrectly identifying the action requested by the speaker and misinterpreting the beliefs and context of the speaker. Another principal source of trouble is the description constructed by the speaker to refer to an actual object in the world. The description can be imprecise, confused, ambiguous or overly specific, it might be interpreted under the wrong context. This leads to difficulty for the listener when figuring out what object is being described, that is, reference identification errors. Such utterances and descriptions are "ill-formed" input. The blame for ill-formedness may lie partly with the speaker and partly with the listener. The speaker may have been sloppy or not taken the hearer into consideration, the listener may be either remiss or unwilling to admit he can't understand the speaker and to ask the speaker for clarification, or may simply feel that he has understood when he in fact has not.

This work is part of an on-going effort to develop a reference identification and plan recognition mechanism that can exhibit more "human-like" tolerance of such utterances. Our goal is to build a more robust system that can handle errorful utterances, and that can be incorporated in existing systems. As a start, we have

concentrated on reference identification. In conversation people use imperfect descriptions to communicate about objects, sometimes their partners succeed in understanding and occasionally they fail. Any computer hoping to play the part of a listener must be capable of taking what the speaker says and either deleting, adapting or clarifying it. We are developing a theory of the use of extensional descriptions that will help explain how people successfully use such imperfect descriptions. We call this the theory of reference miscommunication.

Section 7.2 of this paper highlights some aspects of normal communication and then provides a general discussion on the types of miscommunication that occur in conversation, concentrating primarily on reference problems and motivating many of them with illustrative protocols. Section 7.3 presents possible ways around some of the problems of miscommunication in reference. Motivated there is a (partially completed) implementation of a reference mechanism that attempts to overcome many reference problems.

We are following the task-oriented paradigm of Grosz [15] since it is easy to study (through videotapes), it places the world in front of you (a primarily extensional world), and it limits the discussion while still providing a rich environment for complex descriptions. The task chosen as the target for the system is the assembly of a toy water pump. The water pump is reasonably complex, containing four subassemblies that are built from plastic tubes, nozzles, valves, plungers, and caps that can be screwed or pushed together. A large corpus of dialogues concerning this task was collected by Cohen (see [8, 9, 10]). These dialogues contained instructions from an "expert" to an "apprentice" that explain the assembly of the toy water pump. Both

participants were working to achieve a common goal - the successful assembly of the pump. This domain is rich in perceptual information, allowing for complex descriptions of elements in it The data provide examples of imprecision, confusion, and ambiguity as well as attempts to correct these problems.

The following exchange exemplifies one such situation. Here A is instructing J to assemble part of the water pump. Refer to Figure 13(a) for a picture of the pump. A and J are communicating verbally but neither can see the other. (The bracketed text in the excerpt tells what was actually occurring while each utterance was spoken.) Notice the complexity of the speaker's descriptions and the resultant processing required by the listener. This dialogue illustrates when listeners repair the speaker's description in order to find a referent, when they repair their initial reference choice once they are given more information, and when they fail to choose a proper referent. In Line 7, A describes the two holes on the *BASEVALVE* as "the little hole." J must repair the description, realizing that A doesn't really mean "one" hole but is referring to the "two" holes. J apparently does this since he doesn't complain about A's description and correctly attaches the *BASEVALVE* to the *TUBEBASE* Figure 13(b) shows the configuration of the pump after the *TUBEBASE* is attached to the *MAINTUBE* in Line 10. In Line 13, J interprets "a red plastic piece" to refer to the *NOZZLE*. When A adds the relative clause "that has four gizmos on it," J is forced to drop the *NOZZLE* as the referent and to select the *SLIDEVALVE*. In Lines 17 and 18, A's description "the other--the open part of the main tube, the lower valve" is ambiguous, and J selects the wrong site, namely the *TUBEBASE*, in which to insert the *SLIDEVALVE*. Since the *SLIDEVALVE* fits, J doesn't detect any trouble. Lines 20 and 21 keep J from thinking that something is wrong because the part fits loosely. In Lines

27 and 28. J indicates that A did not give him enough information to perform the requested action. In Line 30, J further compounds the error in Line 18 by putting the *SPOUT* on the *TUBEBASE*.

Excerpt 1 (Telephone)

A 1. Now there s a blue cap
[J grabs the TUBEBASE]
2. that has two little teeth sticking
3. out of the bottom of it.

J 4. Yeah.

A 5. Okay. On that take the
6. bright shocking pink piece of plastic
[J puts down MAINTUBE and takes BASEVALVE]
7. and stick the little hole over the teeth.
[J starts to install the BASEVALVE, backs off, looks at it again and then goes ahead and installs it]

J 8. Okay.

A 9. Now screw that blue cap onto
10. the bottom of the main tube
[J screws TUBEBASE onto MAINTUBE]

J 11. Okay.

A 12. Now, there's a--
13. a red plastic piece
[J starts for NOZZLE]
14. that has four gizmos on it.
[J switches to SLIDEVALVE]

J 15. Yes

A 16. Okay. Put the ungizmoed end in the uh

17. the other--the open

18. part of the main tube, the lower valve.

[J puts SLIDEVALVE into hole in TUBEBASE, but A meant OUTLET2 of MAINTUBE]

J 19 All right

A 20 It just fits loosely It doesn't
21 have to fit right Okay, then take
22 the clear plastic elbow joint.
[J takes SPOUT]

J 23 All right

A 24 And put it over the bottom opening, too.
[J tries installing SPOUT on TUBEBASE]

J 25 Okay.

A 26 Okay. Now, take the--

J 27 Which end am I supposed to put it over?
28 Do you know?

A 29 Put the--put the--the big end--

30 the big end over it
[J pushes big end of SPOUT on TUBEBASE, twisting it to force it on]

7.2 Miscommunication

People must and do manage to resolve lots of (potential) miscommunication in everyday conversation. Much of it is resolved subconsciously - with the listener unaware that anything is wrong. Other miscommunication is resolved with the listener actively deleting or replacing information in the speaker's utterance until it fits the current context. Sometimes this resolution is postponed until the questionable part of the utterance is actually needed. Still, when all these fail, the listener can ask the speaker to clarify what was said.

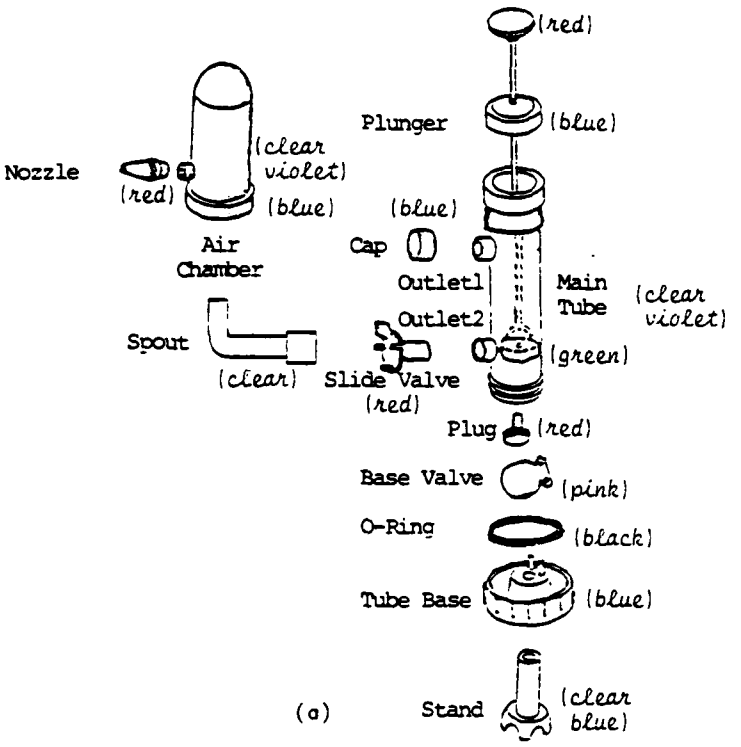


FIG. 13 THE TOY WATER PUMP

There are many aspects of an utterance that the listener can become confused about and that can lead to miscommunication. The listener can become confused about what the speaker intends for the referents, the actions, and the goals described by the utterance. Confusions often appear to result from conflict between the current state of the conversation (the context or focus [15, 24, 31]), the overall goal of the speaker [7], or the manner in which the speaker presented the information. However, when the listener steps back and is able to discover what kind of confusion is occurring, then the confusion can quite possibly be resolved.

7.2.1 Causes of Miscommunication

This section attempts to motivate a paradigm for the kinds of conversation that we studied and tries to point out places in the paradigm that leave room for miscommunication

7.2.1.1 Effects of the Structure of Task-Oriented Dialogues

Task-oriented conversations have a specific goal to be achieved. the performance of a task (e.g., [15]). The participants in the dialogue can have the same skill level and they can simply work together to accomplish the task, or one of them, the expert, could know more and could direct the other, the apprentice, to perform the task. We have concentrated primarily on the latter case - due to the protocols that we examined - but many of our observations can be generalized to the former case, too. We will refer to this as the apprentice-expert domain.

The viewpoints of the expert and apprentice differ greatly in apprentice-expert exchanges. The expert, having an understanding of the functionality of the elements in the task, has more of a feel for how the elements work together, how they go together, and how the individual elements can be used. The apprentice normally has no such knowledge and must base his decisions on perceptual features such as shape [16]

The structure of the task affects the structure of the dialogue [15], particularly through the center of attention of the expert and apprentice. This is the phenomenon called focus [15, 24, 31], which, in task-oriented dialogues is a very real and operational thing (e.g., focus is used in resolving anaphoric references). Shifts in

focus correspond directly to the task, its subtasks, the objects in a task and the subpieces of each object. Focus and focus shifts are governed by many rules [15, 24, 31]. Confusion may result when expected shifts do not take place. For example, if the expert changes focus to an object but never discusses its subpieces (such as an obvious attachment surface) or never bothers to talk about the object reasonably soon after its introduction (i.e., between the time of its introduction and its use, without digressing in a well-structured way in between (see [24])), then the apprentice may become confused, leaving him ripe for miscommunication. The reverse influence between focus and objects can lead to trouble, too. A shift in focus by the expert that does not have a manifestation in the apprentice's world will also perplex the apprentice.

Focus also influences how descriptions are formed [16, 3]. The level of detail required in a description depends directly on the elements currently highlighted by the focus. If the object to be described is similar to other elements in focus, the expert must be more specific in the formulation of the description or may consider shifting focus away from the possibly ambiguous objects to one where the ambiguity won't occur.

7.2.1.2 Influences on Miscommunication

Discrepancies in knowledge between the speaker and listener can cause miscommunication. These disagreements can occur because the listener does not bring sufficient knowledge to the task. They also can occur because the speaker fails to convey enough information in each utterance to bring the listener up to a level of knowledge sufficient to perform the task (that knowledge becomes shared or mutually

believed knowledge [6, 23, 17, 22]). For example, differences between speaker and listener, such as what each believes about the other, can lead to false assumptions that each may use when interpreting the others utterances. Knowledge differences, though, also provide a means to help detect miscommunication. For example, a listener's knowledge about the world in which the task is taking place can provide a way of checking whether or not a speaker s utterance is realistic or not.

Knowledge the Listener Brings to the Task

In apprentice-expert dialogues such as those in the water pump domain, the knowledge brought to the task by a naive apprentice is limited to four principal areas. (1) language abilities, (2) perceptual abilities to identify objects, (3) past experience and knowledge in assembling objects, and (4) the ability to perform trial-and-error tests in the real world The language abilities of the apprentice allow him to follow the flow of information provided by the expert in his utterances and descriptions. Syntactic, semantic and pragmatic knowledge compose this knowledge about language. A more detailed description of these knowledge sources can be found in [27, 13, 33, 14].

Perceptual abilities include the recognition of physical features of an object such as size, shape, color, location, composition and transparency. The fineness of each category's partitioning varies among individuals. For example, some people know more color values than others. An expert, thus, must use only basic level descriptions in each category until the apprentice demonstrates a broader knowledge or the expert can familiarize the apprentice with other values.

The past experience someone has with objects provides a method for the expert

to tie a description down to a common point of view. If an object has a familiar name, the expert can refer to it by that name. The expert can also refer by making analogies to everyday objects as a model for the apprentice in his selection of a referent. The analogies can be through the shapes or functions of everyday objects. The same holds true for actions – past experience makes it easier for the expert to describe an action to the apprentice.

Finally, the apprentice brings to a task the ability to perform simple tests. He can experiment to determine whether two pieces can be attached. In the water pump domain, attachment is performed by pushing, twisting or screwing one object into or onto another. During and after the attachment process, one can determine how good a fit is by noting the compatibility of the shapes of the attaching surfaces (and this can be used to align the surfaces) and by checking the snugness of the fit once the objects are attached.

The Knowledge Transferred in an Utterance

At least two kinds of knowledge are conveyed in an utterance. For this paper we will focus on task knowledge [15] and communicative knowledge [29, 7, 24, 1, 28, 2, 19]. Task knowledge is knowledge about the specific domain. It has three aspects in the water pump domain. (1) the objects, the set of parts available to accomplish the task (the real world), (2) the actions, the set of physical actions available to the listener, and (3) instructions linking objects and actions together to achieve some goal. Communicative knowledge consists of speech acts, communicative goals, and communicative actions. Speech acts are underlying forms that are performed by the speaker in making an utterance (e.g., REQUEST, INFORM) [29, 7, 1]. Communicative goals reflect the structure of the discourse (e.g.,

setting up a topic, clarifying, or adding more information [2]) and express how the utterance is to be understood and hence how the task it examines is used. A communicative act is a way of accomplishing the communicative goal that one wants to convey (e.g., communicate the goal, communicate the object's description, communicate the action). Only some of the possible communicative acts may be reasonable at any one time to accomplish the current communicative goal [25, 2, 19].

Trouble in communication can occur due to the way the information was transferred (i.e., communicative knowledge) or the content of what was transferred (i.e., task knowledge). Problems can occur with the task knowledge when the speaker is unaware that (1) the listener has a different view of the task, (2) the listener is considering a different subset of objects, or (3) the listener is considering a different subset of actions. Difficulties with communicative knowledge are also possible. The speaker may use the wrong speech act (e.g., utters something (inadvertently) that would be conventionally interpreted as an INFORM when meant as a REQUEST) or the listener errs when interpreting the speaker's intention (e.g., the speaker may be INFORMing the listener that the blue cap fits around the end of the tube but the listener might interpret the utterance as a REQUEST to actually place the cap around the end of the tube). In both cases it is the effect of the speech act that causes the trouble since it influences what the listener will do with what was said (i.e., what are the proper responses). Finally, communicative knowledge can cause mistakes and confusion if the listener and speaker differ on the communicative goal (e.g., the listener might think the speaker is clarifying previous information when, in fact, the speaker is adding new information). They will feel they are communicating at cross *purposes* – leading to frustration.

7.2.2 Consequences of Miscommunication

In this section we will make it clear that people do miscommunicate and yet they often manage to fix things. We will look at specific forms of miscommunication and describe ways to detect them. We will highlight relationships between different miscommunication problems but won't necessarily demonstrate ways to resolve each of them.

7.2.2.1 Instances of Miscommunication

There are many ways hearers can get confused during a conversation. Figure 14 outlines some of them that were derived from analyzing the water pump protocols. This section defines and illustrates many of them through numerous excerpts. Each excerpt is marked in parentheses to show what form of communication was used (see [8]). Each bracketed portion of the excerpt explains what was occurring at that point in the dialogue. The confusions themselves, coupled with the sketch at the end of this section on how to recognize when one of *them is occurring*, provides motivation for the use of the algorithm outlined in Section 7.3 as a means for repairing communication problems. We will only discuss referent confusion in this paper. The other forms of confusion – Action, Goal, and Cognitive Load – are described in [12, 14].

Referent confusion occurs when the listener is unable to correctly determine what the speaker is referring to with a particular description.[23] It occurs when the descriptions in the utterance are ambiguous or imprecise, when there is confusion

---

[23] See [35, 15, 34, 31, 25, 13, 14] for discussions on the identification of referents.

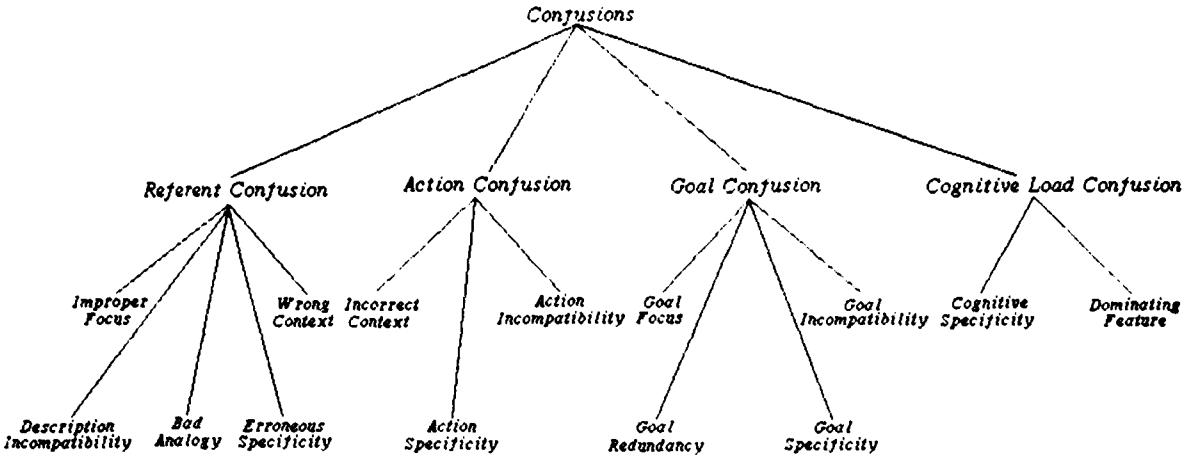


FIG. 14 A TAXONOMY OF CONFUSIONS

between the speaker and listener about what the current focus or context is, or when the descriptions in the utterance are either incorrect or incompatible with the current or global context

Erroneous Specificity

Ambiguous (and, thus, imprecise) descriptions can cause confusion about the referent. Excerpt 2 below illustrates a case where the speaker's description is underspecified - it does not provide enough detail to prune the set of possible referents down to one.

Excerpt 2 (Face-to-Face)

S. 1. And now take the little red
2. peg.

[P takes PLUG]

3. Yes.

4. and place it in the hole at the
5. green end.

[P starts to put PLUG into OUTLET2 of MAINTUBE]

6. no

7. the--in the green thing

[P puts PLUG into green part of PLUNGER]

P. 8. Okay

In Line 4 and 5, S describes the location to place a peg into a hole by giving spatial information. Since the location is given relative to another location by "in the hole at the green end", it defines a region where the peg might go instead of a specific location. In this particular case, there are three possible holes to choose from that are near the green end. The listener chooses one - the wrong one - and inserts the peg into it. Because this dialogue took place face to face, S is able to correct the ambiguity in Lines 6 and 7.

A speaker's description can be imprecise in several possible ways. (1) It may contain features that do not readily apply in the domain. In Line 3, Excerpt 3, the feature "funny" has no relevance to the listener. It is not until A provides a fuller description in Lines 5 to 8 that E is able to select the proper piece. (2) It may use a vague head noun coupled with few or no feature values (and context alone does not necessarily suffice to distinguish the object). In Excerpt 4, Line 9, "attachment" is vague because all objects in the domain are attachable parts. The expert's use of "attachment" was most likely to signal the action the apprentice can expect to take next. The use of the feature value "clear" provides little benefit either because three clear, unused parts exist. The size descriptor "little" prunes this set of possible

referents down to two contenders. (3) Enough feature values are provided but at least one value is too vague leading to trouble. In Excerpt 5, Line 3, the use of the attribute value "rounded" to describe the shape does not sufficiently reduce the set of four possible referents (though, in this particular instance, A correctly identifies it) because the term is applicable to numerous parts in the domain. A more precise shape descriptor such as "bell-shaped" or "cylindrical" would have been more beneficial to the listener.

Excerpt 3 (Telephone)

E 1. All right.

2. Now

3. There's another funny little
4. red thing, a

[A is confused, examines both NOZZLE and SLIDEVALVE]

5. little teeny red thing that's
6. some--should be somewhere on
7. the desk, that has um--there's
8. like teeth on one end

[E takes SLIDEVALVE]

A 9. Okay.

E 10. It's a funny-loo--hollow.
11. hollow projection on one end
12. and then teeth on the other

Excerpt 4 (Teletype)

A 1. take the red thing with the
2. prongs on it

3. and fit it onto the other hole
4. of the cylinder

5. so that the prongs are
6. sticking out

R. 7. ok

A. 8. now take the clear little
9. attachment

10. and put on the hole where you
11. just put the red cap on

12. make sure it points
13. upward

R. 14. ok

Excerpt 5 (Teletype)

S 1. Ok.

2. put the red nozzle on the outlet
3. of the rounded clear chamber

4. ok?

A. 5. got it.

Improper Focus

Focus confusion can occur when the speaker sets up one focus and then proceeds with another one without letting the listener know of the switch (i.e., a focus shift occurs without any indication). An opposite phenomenon can also happen – the listener may feel that a focus shift has taken place when the speaker actually never intended one. These really are very similar – one is viewed more strongly from the perspective of the speaker and the other from the listener.

Excerpt 6 below illustrates an instance of the first type of focus confusion. In the excerpt, the speaker (S) shifts focus without notifying the listener (P) of the switch. S provides in Lines 1 to 17 instructions for P to attach the *CAP* and the *SPOUT* to outlets *OUTLET1* and *OUTLET2*, respectively, on the *MAINTUBE*. Upon P's successful completion of these attachments, S switches focus in Lines 18 to 21 to the *TUBEBASE* assembly and requests P to screw it on to the bottom of the *MAINTUBE*. While P completes the task, S realizes she left out a step in the assembly – the placement of the *SLIDEVALVE* into *OUTLET2* of the *MAINTUBE* before the *SPOUT* is placed over the same outlet. S attempts to correct her mistake by requesting P to remove "the plas"[24] piece in Lines 22 and 23. Since S never indicated a shift in focus from the *TUBEBASE* back to the *SPOUT*, P interprets "the plas" to refer to the *TUBEBASE*.

Excerpt 6 (Face-to-Face)

[P holding TUBEBASE]

S 1 And place
2 the blue cap that's left
[P takes CAP]
3 on the side holes that are
4 on the cylinder.
[P lays down TUBEBASE]
5 the side hole that is farthest
6 from the green end
[P puts CAP on OUTLET1 of MAINTUBE]

P 7 Okay

---

[24]The whole word here is "plastic." People in general tend to be good at proceeding before hearing the whole utterance or even the whole word.

S. 8. And take the nozzle-looking
9. piece.
[P grabs NOZZLE]

10. no

11. I mean the clear plastic one.
[P takes SPOUT]

12. and place it on the other hole
[P identifies OUTLET2 of MAINTUBE]
13. that s left.

14. so that nozzle points away
15. from the
[P installs SPOUT on OUTLET2 of MAINTUBE]

16. right.

P 17. Okay.

S. 18. Now

19. take the

20. cap base thing
[P takes TUBEBASE]
21. and screw it onto the bottom.
[P screws TUBEBASE on MAINTUBE]
22. ooops.
[S realizes she has forgotten to have P put SLIDEVALVE into OUTLET2 of MAINTUBE]
23. un-undo the plas
[P starts to take TUBEBASE off MAINTUBE]

24. no

25. the clear plastic thing that I
26. told you to put on
[P removes SPOUT]

27. sorry

28. And place the little red thing

[P takes SLIDEVALVE]

29. in there first.

[P inserts SLIDEVALVE into OUTLET2 of MAINTUBE]

30. it fits loosely in there

Excerpt 7 below demonstrates the latter type of focus confusion that occurs when the speaker (S) sets up one focus - the *MAINTUBE*, which is the correct focus in this case - but then proceeds in such a manner that the listener (J) thinks a focus shift to another piece, the *TUBEBASE*, has occurred. Thus, Line 15 refers to "the lower side hole in the *MAINTUBE*" for S and "the hole in the *TUBEBASE*" for J. J has no way of realizing that he has focussed incorrectly unless the description as he interprets it doesn't have a real world correlate (here something does satisfy the description so J doesn't sense any problem) or if, later in the exchange, a conflict arises due to the mistake (e.g., a requested action can not be performed). In Line 31, J inserts a piece into the wrong hole because of the misunderstanding in Line 15. Line 31 hints that J may have become suspicious that an ambiguity existed but since the task was successfully completed (i.e., the red piece fit into the hole in the base), and since S did not provide any clarification, he assumed he was correct.

Excerpt 7 (Telephone)

S. 1. Um now.
2. Now we're getting a little
3. more difficult.

J 4. (laughs)

S 5. Pick out the large air tube

[J picks up STAND]

6. that has the plunger in it.

[J puts down STAND, takes PLUNGER/MAINTUBE assembly]

J. 7 Okay.

S. 8 And set it on its base.

[J puts down MAINTUBE, standing vertically, on the TABLE]

9 which is blue now.
10 right?

[J has shifted focus to the TUBEBASE]

J. 11 Yeah.

S 12 Base is blue
13 Okay.
14 Now
15 You've got a bottom hole still
16 to be filled.
17 correct?

J 18 Yeah

[J answers this with MAINTUBE still sitting on the TABLE; he shows no indication of what hole he thinks is meant – the one on the MAINTUBE, OUTLET2, or the one in the TUBEBASE]

S 19 Okay.
20 You have one red piece
21 remaining?

[J picks up MAINTUBE assembly and looks at TUBEBASE, rotating the MAINTUBE so that TUBEBASE is pointed up, and sees the hole in it; he then looks at the SLIDEVALVE]

J. 22 Yeah.

S. 23 Okay.
24 Take that red piece

[J takes SLIDEVALVE]

25 It's got four little feet on
26 it?

J. 27 Yeah.

S. 28 And put the small end into

29. that hole on the air tube--

30 on the big tube.

J 31. On the very bottom?

[J starts to put it into the bottom hole of TUBEBASE - though he indicates he is unsure of himself]

S 32. On the bottom.
33. Yes.

Misfocus can also occur when the speaker inadvertently fails to distinguish the proper focus because he did not notice a possible ambiguity, or when, through no fault of the speaker, the listener just fails to recognize a switch in focus indicated by the speaker Excerpt 7 above is an example of the first type because S failed to notice that an ambiguity existed since he never explicitly brought the *TUBEBASE* either into or out of focus He just assumed that J had the same perspective as him - a perspective in which no ambiguity occurred.

Wrong Context

Context differs from focus. The context of a portion of a conversation is concerned with the point of the discussion in that fragment and with the set of objects relevant to that discussion, though not attended to currently Focus pertains to the elements which are currently being attended to in the context. For example, two people can share the same context but have different focus assignments within it - we're both talking about the water pump but you're describing the *MAINTUBE* and I'm describing the *AIRCHAMBER*. Alternatively, we could just be using different contexts - I think you're talking about taking the pump apart but you're talking about replacing the pump with new parts - in both cases we may be sharing the same

focus - the pump - but our contexts are totally off from one another.[25] The kinds of misunderstandings that can occur because of context problems are similar to those for focus problems. (1) the speaker might set up or be in one context for a discussion and then proceed in another one without effectively letting the listener know of the change, (2) the listener may feel a change in context has taken place when in fact the speaker never intended one, or (3) the listener fails to recognize an indicated context switch by the speaker. Context affects reference because it helps define the set of available objects that are possible contenders for the referent of the speaker's descriptions. If the contexts of the speaker and listener differ, then misreference might result.

Bad Analogy

An analogy (see [11] for a discussion on analogies) is a useful way to help describe an object by attempting to be more precise by using shared past experience and knowledge - especially shape and functional information. If that past experience or knowledge doesn't contain the information the speaker assumes it does or isn't there, then trouble occurs. Thus, one more way referent confusion can occur is by describing an object using a poor analogy. An analogy used to describe an object might not be specific enough - confusing the listener because several pieces might conform to the analogy or, in fact, none at all appear to fit because discovering a mapping between the analogous object and some piece in the environment is too difficult. In Excerpt 6, J at first has trouble correctly satisfying A's functional analogy "stopper" in "the big blue stopper", but finally selects what he considers to be the closest match to "stopper".

---

[25] Grosz [15, 16] would describe this as a difference in "task plans" while Reichman [24, 25] would say that the "communicative goals" differed.

Excerpt 8 (Telephone)

A 1. Okay. Now,

2. take the big blue
3. stopper that's laying around
[J grabs AIRCHAMBER]

4. ... and take the black
5. ring--

J 6. The big blue stopper?
[J is confused and tries to communicate it to A; he is holding the AIRCHAMBER here]

A 7. Yeah.

8. the big blue stopper

9. and the black ring
[J drops AIRCHAMBER and takes the O-RING and the TUBEBASE]

In other cases it might be too specific - confusing the listener because none of the available referents appear to fit it. In Line 8 of Excerpt 6, "nozzle-looking" forms a poor shape analogy because the object being referred to actually is an elbow-shaped spout. The "nozzle-looking" part of the description convinced the listener that what he was looking for was something specific like a nozzle (which is a small spout). Sometimes, when an object is a clear representative of a specified analogy class, the apprentice may become confused, wondering why the expert bothered to form an analogy instead of just directly describing the object as a member of the class. Hence, it would not be surprising if the apprentice ignored the best representative of the class for some less obvious exemplar. Thus, for example, it is

better to say "nozzle" instead of "nozzle-looking." In Excerpt 9, the description "hippopotamus face shape" (a shape analogy) in Lines 2 and 3, and "champagne top" (a shape analogy) in Line 9, are too specific and the listener is unable to easily find something close enough to match either of them. He can't discover a mapping between the object in the analogy and one in the real world.

Excerpt 9 (Audiotape)

M. 1. take the bright pink flat
2. piece of hippopotamus face
3. shape piece of plastic
4. and you notice that the two
5. holes on it

[M is trying to refer to BASEVALVE]

6. match
7. along with the two
8. peg holes on the
9. champagne top sort of
10. looking bottom that had
11. threads on it

[M is trying to refer to TUBEBASE]

Description Incompatibility

Incompatible descriptions can lead to confusion also. A description is incompatible when (1) one or more of the specified conditions, i.e., the feature values, do not satisfy <u>any</u> of the pieces; (2) when one or more specified constraints do not hold (e.g., saying "the <u>loose</u> one" when all objects are tightly attached); or (3) if no <u>one</u> object satisfies <u>all</u> of the features specified in the description. In Lines 7 and 8 of Excerpt 9 above M's use of "the two peg holes" leads to bewilderment for the listener because the described object has no holes in it. M actually meant "two pegs".

7.2.2.2 Detecting Miscommunication

Part of our research has been to examine how a listener discovers the need for a repair of an utterance or a description during communication. The incompatibility of a referent or action is one signal of possible trouble. The appearance of an obstacle that blocks one from achieving a goal is another indication of a problem.

Incompatibility

Two kinds of incompatibility, action or referent, appear in the taxonomy of confusions. The strongest hint that there is a reference problem occurs when the listener finds no real world object to correspond to the speaker's description. This can occur when (1) one or more of the specified feature values in the description are not satisfied by any of the pieces (e.g., saying "the orange cap" when none of the objects are orange), (2) when one or more specified constraints do not hold (e.g., saying "the red plug that fits loosely" when all the red plugs attach tightly), or (3) if no one object satisfies all of the features specified in the description (i.e., there is, for each feature, an object that exhibits the specified feature value, but no one object exhibits all of the values). An action problem is likely if (1) the listener cannot perform the action specified by the speaker because of some obstacle, (2) the listener performs the action but does not arrive at its intended effect (i.e., a specified or default constraint isn't satisfied), or (3) the current action affects a previous action in an adverse way, yet the speaker has given no sign of any importance to this side-effect.

Goal Obstacle

A goal obstacle occurs when a goal (or subgoal) one is trying to achieve is blocked. This blockage can result in confusion for the listener because he did not

expect the speaker to give him tasks that could not be achieved. Often, though, it points out for the listener that some miscommunication has occurred.

Goal Redundancy

Goal redundancy occurs when the requested goal (or subgoal) is already satisfied. In some sense, it is a special kind of goal obstacle where the goal to be fulfilled is blocked because it is already satisfied. It is a simple goal obstacle because nothing has to be done to get around it. However, it can lead to confusion on the part of listeners because they may suspect they misunderstood what the speaker has requested since they wouldn't expect a reasonable speaker to request the performance of an already completed action. It provides a hint that miscommunication has occurred.

7.2.3 Summary

We have attempted to show in the preceding sections that miscommunication occurs often in the real world between human conversants. It seems inevitable that miscommunication will also occur if people and computers cooperate on tasks, and so computers will have to be able to handle such problems. Miscommunication is <u>often</u> resolved subconsciously (possibly in a manner analogous to a relaxation process). Many times, however, it can only be detected when the hearer's expectations and possible interpretations of an utterance do not agree with his perception of the objects and relations in the physical world. In essence, then, there are two kinds of miscommunication – the easy ones that can be resolved instantly and the ones that are <u>actively</u> noticed and that require the hearer to step back and consider past dialogue or to ask for clarification from the speaker.

7.3 Repairing Reference Failures

7.3.1 Introduction

The previous section illustrated how task-oriented natural language interactions in the real world can induce contextually poor utterances. Given all the possibilities for confusion, when confusions do occur, they must be resolved if the task is to be performed. This section explores the problem of fixing reference failures.

Reference identification is a search process where one looks for something that satisfies the speaker's uttered expression. A computational scheme for performing reference identification has evolved (e.g., see [15]). The traditional approach succeeds if a referent is found or fails if no referent is found (see Figure 15(a)). However, we feel that reference identification is more versatile than presented in the traditional approach. Internal negotiation seems to be another part of reference. For example, linguistic descriptions can color a listener's perception of the world. The listener must ask himself whether he can perceive one of the objects in the world the way the speaker described it, or whether his perception overrules finding anything similar to the speaker's description. We are developing an algorithm that captures internal negotiation for certain cases (see Figure 15(b)). It can look for a referent and, if it doesn't find one, it tries to find possible referent candidates that might work, and then loosens the speaker's description where necessary. Thus, the reference process becomes multi-step and resumable. This computational model is more faithful to the data than the traditional model.

One means of making sense of an approximate description is to delete or replace

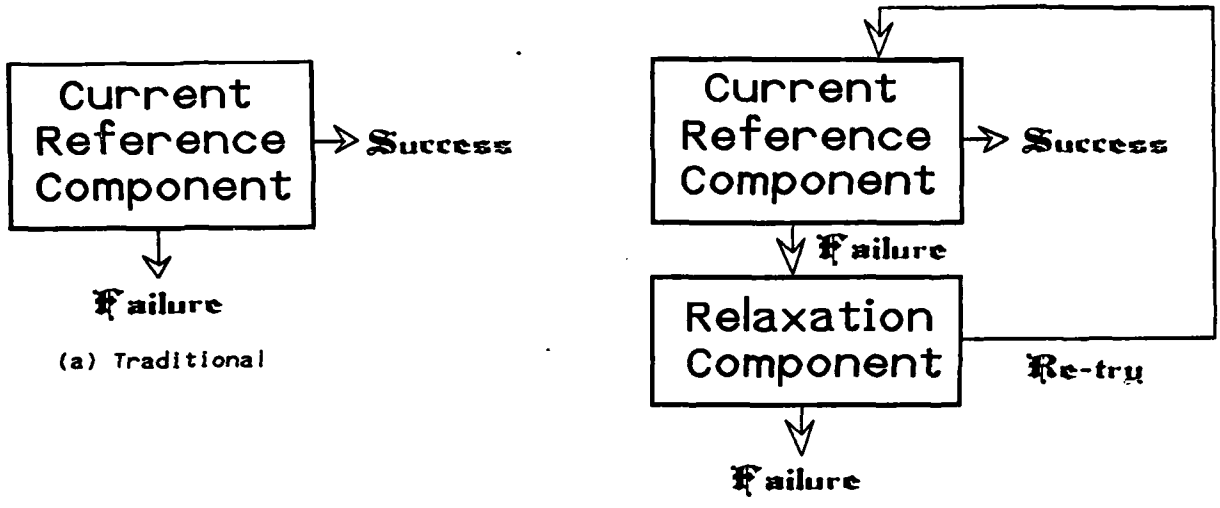


FIG 15 APPROACHES TO REFERENCE IDENTIFICATION

portions of it that don't match objects in the hearer's world In our program we are using relaxation techniques to capture this behavior. Our reference identification module treats descriptions as approximate. It relaxes a description in order to find a referent when the literal content of the description fails to provide the needed information. Relaxation then becomes a form of communication repair (e.g., see [5] for a description on repair theory) that hearers can use The relaxation component of the reference identification module is described below.

When things go wrong during a conversation, people have lots of knowledge that they bring to bear to get around the problem (see [26]). Much of the time the repairs are so natural that we aren't conscious of the knowledge used in making the repairs. At other times, we must make an effort to correct what we have heard, or determine that we need clarification from the speaker. Either repair process involves the use of knowledge about conversation, social conventions and the world around us. We draw primarily on sources of linguistic knowledge, pragmatic knowledge, discourse knowledge, domain knowledge, perceptual knowledge, hierarchical knowledge, and trial

and error knowledge during the repair process. A detailed treatment of these knowledge sources can be found in [13, 33, 14].

7.3.2 The Relaxation Component

When a description fails to denote a referent in the real world properly, it is possible to repair it by a relaxation process that ignores or modifies parts of the description. Since a description can specify many features of an object, the order in which parts of it are relaxed is crucial. There are several kinds of relaxation possible. One can *ignore* a constituent, *replace* it with *something close*, replace it with a related value, or change focus (i.e., consider a different group of objects). This section describes the overall relaxation component that draws on knowledge sources about descriptions and the real world as it tries to relax an errorful description to one for which a referent can be identified.

7.3.2.1 Find a Referent Using a Reference Mechanism

Identifying the referent of a description requires finding an element in the world that corresponds to the speaker's description (where every feature specified in the description is *present* in the *element* in the world but not necessarily vice versa). The initial task of our reference mechanism is to determine whether or not a search of the (taxonomic) knowledge base[26] is necessary. For example, the reference component should not bother searching – unless specifically requested to do so – for a referent for indefinite noun phrases (which usually describe new or hypothetical objects) or extremely vague descriptions (which do not clearly describe an object

---

[26]The knowledge base contains linguistic descriptions and a description of the listener's visual scene itself. In our implementation and algorithms, we assume it is represented in KL-One [4], a system for describing taxonomic knowledge.

because they are composed of imprecise feature values). A number of aspects of discourse pragmatics can be used in that determination (e.g., the use of a deictic in a definite noun phrase, such as "this X" or "the last X", hints that the object was either mentioned previously or that it probably was evoked by some previous reference, and that it is searchable) but we will not examine them here.

Assuming that a search of the knowledge base is considered necessary, then a reference search mechanism is invoked. The search mechanism uses the KL-One Classifier [18] to search[27] the knowledge base taxonomy. The Classifier's purpose is to discover all appropriate subsumption relationships between a newly formed description and all other descriptions in a given taxonomy. With respect to reference, this means that all possible (descriptions of) referents of the description will be subsumed by it after it has been classified into the knowledge base taxonomy. If more than one candidate referent is below (when a description A is subsumed by B, we say A is "below" B) the classified description, then, unless a quantifier in the description specified more than one element, the speaker's description is ambiguous. If exactly one description is below it, then the intended referent is assumed to have been found. Finally, if no referent is found below the classified description, the relaxation component is invoked.

7.3.2.2 Collect Votes For or Against Relaxing the Description

It is necessary to determine whether or not the lack of a referent for a description has to do with the description itself (i.e., reference failure) or outside

---

[27] This search is constrained by a focus mechanism [15, 24, 31].

forces that are causing reference confusion. For example, the problem may be with the flow of the conversation and the speaker's and listener's perspectives on it; it may be due to incorrect attachment of a modifier, it may be due to the action requested, and so on. Pragmatic rules are invoked to decide whether or not the description should be relaxed. These rules will not be discussed here.

7.3.2.3 Perform the Relaxation of the Description

If relaxation is demanded, then the system must (1) find potential referent candidates, (2) determine which features to relax and in what order, and use those ordered features to order the potential candidates with respect to the preferred ordering of features, and (3) determine the proper relaxation techniques to use and apply them to the description

Find Potential Referent Candidates

Before relaxation can take place, potential candidates for referents (which denote elements in the listener's visual scene) must first be found. These candidates are discovered by performing a "walk" in the knowledge base taxonomy in the general vicinity of the speaker's classified description. A KL-One partial matcher is used to determine how close the candidate descriptions found during the walk are to the speaker s description and is used to assign scores to matches. The partial matcher generates a score to represent how well the descriptions match (after first generating scores at the feature level to help determine how the features are to be aligned and how well they match). This score is based on information about KL-One and does not take into account any information about the task domain. The ordering of features and candidates for relaxation described below takes into account the task domain. The set of best descriptions returned by the matcher (as determined by some cutoff score) are selected as referent candidates.

Order the Features and Candidates for Relaxation

At this point the reference system inspects the speaker's description and the candidates, decides which features to relax and in what order,[28] and generates a master ordering of features for relaxation. Once the feature order is created, the reference system uses that ordering to determine the order in which to try relaxing the candidates

Various knowledge sources are consulted to determine the feature ordering for relaxation (see [13, 33, 14]). Each knowledge source produces its own partial ordering of features. The partial orderings are then integrated to form a directed graph. For example, perceptual knowledge may say to relax color. However, if the color value was asserted in a relative clause, linguistic knowledge would rank color lower, i.e., placing it later in the list of things to relax.

Since different knowledge sources generally have different partial orderings of features, these differences lead to a conflict over which features to relax. It is the job of the best candidate algorithm to resolve the disagreements among knowledge sources. It's goal is to order the referent candidates, C_i, so that relaxation is attempted on the best candidates first. Those candidates are the ones that conform best to a proposed feature ordering. To start, the algorithm examines pairs of candidates and the feature orderings from each knowledge source For each candidate

---

[28] Of course, once one particular candidate is selected, then deciding which features to relax is relatively trivial – one simply compares feature by feature between the candidate description (the target) and the speaker's description (the pattern) and notes any discrepancies.

C_i, the algorithm scores the effect of relaxing the speaker s original description to C_i, using the feature ordering from one knowledge source The score reflects the goal of minimizing the number of features relaxed while trying to relax the features that are "earliest" in the feature ordering It repeats its scoring of C_i for each knowledge source, and sums up its scores to form C_i s total score The C_i s are then ordered by that score.

Figure 16 provides a graphic description of this process. A set of objects in the real world are selected by the partial matcher as potential candidates for the referent. These candidates are shown across the top of the figure. The lines on the right side of each box correspond to the set of features that describe that object. The speaker's description is represented in the center of the figure. The set of specified features and their assigned feature value (i.e., the pairs fi-vi) are also shown there. A set of partial orderings are generated that suggest which features in the speaker's description should be relaxed first - one ordering for each knowledge source (shown as "a", "b", and "c" in the figure). These are put together to form a directed graph that represents the possible, reasonable ways to relax the features specified in the speaker s description. Finally, the referent candidates are reordered using the information expressed in the speaker's description and in the directed graph of features.

Once a set of ordered, potential candidates are selected, the relaxation mechanism begins step 3 of relaxation, it tries to find proper relaxation methods to relax the features that have just been ordered (success in finding such methods "justifies" relaxing the description). It stops at the first candidate which is reasonable.

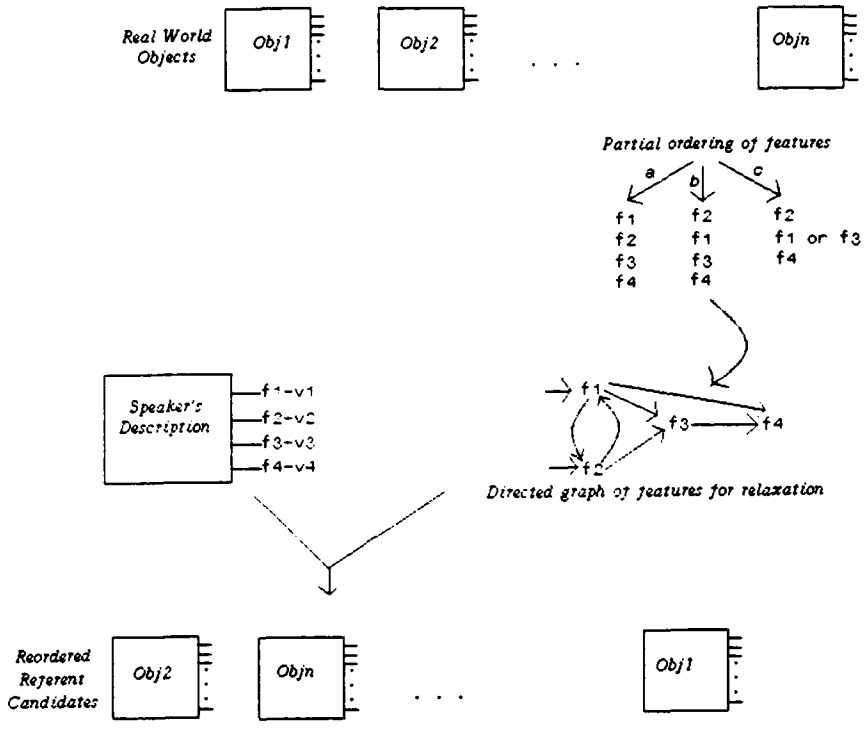


FIG 16. REORDERING REFERENT CANDIDATES

Determine which Relaxation Methods to Apply

Relaxation can take place with many aspects of a speaker's description, with complex relations specified in the description, with individual features of a referent specified by the description, and with the focus of attention in the real world where one attempts to find a match. Complex relations specified in a speaker's description include spatial relations (e.g., "the outlet *near* the *top* of the tube"), comparatives (e.g., "the *larger* tube") and superlatives (e.g., "the *longest* tube"). These can be relaxed. The simpler features of an object (such as size or color) that are specified in the speaker's description are also open to relaxation.

Often the objects in focus in the real world implicitly cause other objects to be

in focus [15, 34]. The subparts of an object in focus, for example, are reasonable candidates for the referent of a failing description and should be checked. At other times, the speaker might attribute features of a subpart of an object to the whole object (e.g., describing a plunger that is composed of a red handle, a metal rod, a blue cap, and a green cup as "the green plunger"). In these cases, the relaxation mechanism utilizes the part-whole relation in object descriptions to suggest a way to relax the speaker's description.

Relaxation of a description has a few global strategies that can be followed for each part of the description. (1) drop the errorful feature value from the description altogether, (2) weaken or tighten the feature value but keep its new value close to the specified one, or (3) try some other feature value

These strategies are realized through a set of procedures (or *relaxation methods*) that are organized hierarchically Each procedure is an expert at relaxing its particular type of feature For example, a Generate-Similar-Feature-Values procedure is composed of procedures like Generate-Similar-Shape-Values and Generate-Similar-Size-Values. Each of those procedures are further divided into specialists that first attempt to relax the feature value to one "near" the current one (e.g., one would prefer to first relax the color "red" to "pink" before relaxing it to "blue") and then, if that fails, to try relaxing it to any of the other possible values. If those fail, the feature would simply be ignored.

7.3.3 An Example on Handling a Misreference

This section provides a sketch of how a referent identification system could try to handle a misreference using the scheme outlined in the previous section. For the purposes of this example, assume that the water pump objects currently in focus include the *CAP*, the *MAINTUBE*, the *AIRCHAMBER* and the *STAND* (see Figure 13(a) for a picture of these parts). Assume also that the speaker tries to describe two of the objects. "...two devices that are clear plastic. One of them has two openings on the outside with threads on the end, and its about five inches long. The other one is a rounded pi[illegible]e with a turquoise base on it. Both are tubular. The rounded piece fits loosely over [illegible]. The reference system can find a unique referent for the first object but not for the second. The relaxation algorithm will be shown below to reduce the set of referent candidates for the second description down to two. It, then, requires the system/listener to try out those candidates to determine if one, or both, fits loosely. The protocols exhibit a similar result when the listener uses "fits loosely" to get the correct referent.

Figure 17 provides a simplified and linearized view of the actual KL-One representation of the speaker's descriptions after they have been parsed and semantically interpreted. A representation of each of the water pump objects that are currently under consideration is presented in Figure 18. Each provides a physical description of the object - in terms of its dimensions, the basic 3-D shapes composing it, and its physical features - and a basic functional description of the object. The first entry in each representation in Figure 18 (that entry is shown in uppercase) defines the basic kind of entity being described (e.g.: "TUBE" means that the object

being described is some kind of tube). The words in mixed case refer to the names of features and the words in uppercase refer to possible fillers of those features from things in the water pump world. The "Subpart" feature provides a place for an embedded description of an object that is a subpart of a parent object. Such subparts can be referred to on their own or as part of the parent object. The "Orientation" feature, used in the representations in Figure 18, provides a rotation and translation of the object from some standard orientation to the object's current orientation in 3-D space. The standard orientation provides a way to define relative positions such as "top," "bottom," or "side."

```
Descr1:
 (DEVICE (Transparency CLEAR)
         (Composition PLASTIC)
         (Subpart (OPENING))
         (Subpart (OPENING))
         (Subpart (THREADS (Rel-Position END)))
         (Dimensions (Length 5.0))
         (Analogical-Shape TUBULAR))
```

```
Descr2:
 (FIT-INTO (Outer (DEVICE (Transparency CLEAR)
                          (Composition PLASTIC)
                          (Shape ROUND)
                          (Analogical-Shape TUBULAR)
                          (Subpart (BASE (Color TURQUOISE)))))
           (Inner ....)
           (FitCondition LOOSE))
```

FIG 17 THE SPEAKER'S DESCRIPTIONS

The first step in the reference process is the actual search for a referent in the knowledge base. The KL-One Classifier compares the features specified in the speaker's descriptions (**Descr1** and the "Outer" feature of **Descr2** in Figure 17) with the features specified for each element in the KL-One taxonomy that corresponds to one of the current objects of interest in the real world. Notice that some features are directly comparable. For example, the "Transparency" feature of **Descr1** and the "Transparency" feature of *MAINTUBE* are both equal to "CLEAR." Other features require further processing before they can be compared. The OPENING value of "Subpart" in **Descr1** is thought of primarily as a 2-D cross-section (such as a "hole"),

while the CYLINDER subparts of *MAINTUBE* are viewed as (3-D) cylinders that have the "Function" of being outlets, i.e., OUTLET-ATTACHMENT-POINTS. To compare OPENING and CYLINDER, the inference must be made that both things can describe the same thing (similar inferences are developed in [20]). One way this inference can occur is by recursively examining the subparts of *MAINTUBE* with the partial matcher until the cylinders are examined at the 2-D level. At that level, an end of the cylinder will be defined as an OPENING. With that examination, the *MAINTUBE* can be seen as described by **Descr1**.

Descr2 presents different problems. **Descr2** refers to an object that is supposed to have a subpart that is TURQUOISE. The Classifier determines that **Descr2** could not describe either the *CAP* or *STAND* because both are BLUE. It also could not describe the *MAINTUBE*[29] or *AIR CHAMBER* since each has subparts that are either VIOLET or BLUE. The Classifier places **Descr2** as best it can in the taxonomy, showing no connections between it and any of the objects currently in focus. At this point, a probable misreference is noted. The reference mechanism now tries to find potential referent candidates, using the exploration routine described in Section 7.3.2.3, by examining the elements closest to **Descr2** in the taxonomy and using the partial matcher to score how close each element is to **Descr2**. The matcher determines *MAINTUBE*, *STAND*, and *AIR CHAMBER* as reasonable candidates by aligning and comparing their features to **Descr2**.

---

[29]Since **Descr1** refers to *MAINTUBE*, *MAINTUBE* could be dropped as a potential referent candidate for **Descr2**. We will, however, leave it as a potential candidate to make this example more complex.

```
   (CAP  (Color BLUE)
         (Composition PLASTIC)
CAP      (Transparency OPAQUE)
         (Dimensions (Length .25) (Diameter .5))
         (Orientation (Rotation (0 0 0.0 90.0))
                      (Translation (0 0 0.0 0.0))))
```

```
   (TUBE (Color VIOLET)
         (Composition PLASTIC)
         (Transparency CLEAR)
         (Dimensions (Length 4.125))
         (Subpart (CYLINDER (Dimensions (Length .25) (Diameter 1.125))
                            (Orientation (Rotation (0.0 0.0 0.0))
          Lip                           (Translation (0.0 0.0 3.75)))
                            (Function OUTLET-ATTACHMENT-POINT)))
         (Subpart (CYLINDER (Dimensions (Length 3.5) (Diameter 1.0))
MAIN      TubeBody          (Orientation (Rotation (0.0 0.0 0.0))
TUBE                                    (Translation (0.0 0.0 .25)))))
         (Subpart (CYLINDER (Dimensions (Length .25) (Diameter 1.125))
                            (Orientation (Rotation (0.0 0.0 0.0))
          Threads           (Translation (0.0 0.0 0.0)))
                            (Function THREADED-ATTACHMENT-POINT)))
         (Subpart (CYLINDER (Dimensions (Length .375) (Diameter .5))
                            (Orientation (Rotation (0.0 0.0 90.0))
          Outlet1           (Translation (0.0 .5 3.00)))
                            (Function OUTLET-ATTACHMENT-POINT)))
         (Subpart (CYLINDER (Dimensions (Length .375) (Diameter .5))
                            (Orientation (Rotation (0.0 0.0 90.0))
          Outlet2           (Translation (0.0 .5 .625))
                            (Function OUTLET-ATTACHMENT-POINT))))
```

```
    (CONTAINER (Dimensions (LENGTH 2.75))
               (Composition PLASTIC)
               (Subpart (HEMISPHERE (Color VIOLET)
                                    (Transparency CLEAR)
                Chamber             (Dimensions (Diameter 1.0))
                Top                 (Orientation (Rotation (0.0 0.0 0.0))
                                                 (Translation (0.0 0.0 2.25)))))
               (Subpart (CYLINDER (Color VIOLET)
                                  (Transparency CLEAR)
                Chamber           (Dimensions (Length 1.0) (Diameter 2.25))
                Body              (Orientation (Rotation (0 0 0.0 0.0))
                                               (Translation (0.0 0.0 .375)))))
               (Subpart (CYLINDER (Color BLUE)
                                  (Transparency OPAQUE)
                                  (Dimensions (Length .375) (Diameter 1.25))
AIR                               (Orientation (Rotation (0.0 0.0 0.0))
CHAMBER         Chamber                        (Translation (0.0 0.0 0.0)))
                Bottom            (Function CAP OUTLET-ATTACHMENT-POINT)
                                  (Subpart (CYLINDER (Color BLUE)
                                                     (Dimensions (Length .375)
                                                                 (Diameter .5))
                                                     (Orientation
                                                       (Rotation (0.0 0.0 0.0))
                                                       (Translation (0.0 0.0 0.0)))
                                                     (Function
                                                        OUTLET-ATTACHMENT-POINT)))))
               (Subpart (CYLINDER (Color VIOLET)
                                  (Transparency CLEAR)
                Chamber           (Dimensions (Length .5) (Diameter .375))
                Outlet            (Orientation (Rotation (0.0 0.0 90.0))
                                               (Translation (.625 .625 .625)))
                                  (Function OUTLET-ATTACHMENT-POINT))))
```

FIG. 18. THE OBJECTS IN FOCUS

```
(TUBE (Dimensions (Length 2.75))
      (Composition PLASTIC)
      (Subpart (CYLINDER (Color BLUE)
                         (Transparency CLEAR)
       Top               (Dimensions (Length 2.25) (Diameter .375))
                         (Orientation (Rotation (0 0 0.0 0.0))
STAND                                 (Translation (.5 0.0 .375)))
                         (Function OUTLET-ATTACHMENT-POINT)))
      (Subpart (CYLINDER (Color BLUE)
                         (Transparency CLEAR)
       Base              (Dimensions (Length .375) (Diameter 1.0))
                         (Orientation (Rotation (0.0 0.0 0.0))
                                      (Translation (0.0 0.0 0.0)))
                         (Function OUTLET-ATTACHMENT-POINT))))
```

FIGURE 18. CONCLUDED

Scoring **Descr2** to *MAINTUBE*.

- a TUBE is a kind of DEVICE.
- the Transparency of each is CLEAR.
- the Composition of each is PLASTIC.
- a TUBE implies Analogical-Shape TUBULAR, which implies Shape CYLINDRICAL, which is a kind of Shape ROUND.
- the recursive partial matching of subparts. A BASE is viewed as a kind of BOTTOM. *Therefore*, BASE in **Descr2** could match to the subpart in *MAINTUBE* that has a Translation of (0.0 0.0 0.0) - i.e., *Threads* of *MAINTUBE*. However, they mismatch since color TURQUOISE in **Descr2** differs from color VIOLET of *MAINTUBE*

Scoring **Descr2** to *STAND*.

- a TUBE is a kind of DEVICE.
- the Transparency of each is CLEAR.
- the Composition of each is PLASTIC.
- a TUBE implies Analogical-Shape TUBULAR, which implies Shape CYLINDRICAL, which is a kind of Shape ROUND.
- the recursive partial matching of subparts: BASE in **Descr2** could match to the subpart in *STAND* that has a Translation of (0.0 0.0 0.0) - i.e., *Base* of *STAND*. However, they mismatch since color TURQUOISE in **Descr2** differs from color BLUE of *STAND*.

Scoring **Descr2** to *AIR CHAMBER*.

- a CONTAINER is a kind of DEVICE.
- the Transparency of **Descr2**, CLEAR, matches the Transparency of *ChamberTop*, *ChamberOutlet* and *ChamberBody* of *AIR CHAMBER* but mismatches the Transparency of *ChamberBottom* of *AIR CHAMBER*. Therefore, the partial match is uncertain.
- the Composition of each is PLASTIC.
- the subparts of *AIR CHAMBER* have Shape HEMISPHERICAL and CYLINDRICAL which are each a kind of Shape ROUND.
- the recursive partial matching of subparts. BASE in **Descr2** could match to the subpart in *AIR CHAMBER* that has a translation of (0.0 0.0 0.0) - i.e., *ChamberBottom* of *AIR CHAMBER*. However, they mismatch since color TURQUOISE in **Descr2** differs from color BLUE of *AIR CHAMBER*.

The above analysis using a partial matcher provides no clear winner since the differences are so close causing the scores generated for the candidates to be almost exactly the same. Knowledge about the actual domain (i.e., the water pump) and about language (i.e., the speaker's description itself) can be used to order the features specified in **Descr2** for relaxation. Linguistic analysis of **Descr2**, "... are clear plastic ... a rounded piece with a turquoise base ... Both are tubular ... fits loosely over ...." tells us that the features were specified using the following modifiers.

- Adjective (Shape ROUND)
- Prepositional Phrase, (Subpart (*BASE* (Color *TURQUOISE*)))
- Predicate Complement. (Transparency CLEAR), (Composition PLASTIC), (Analogical-Shape TUBULAR), (Fit LOOSE)

Observations from the protocols (as described by the rules developed in [14]) has shown that people tend to relax first features specified as adjectives, then as prepositional phrases and finally as relative clauses or predicate complements. This suggests relaxation of **Descr2** in the order.

{Shape} < {Color,Subpart} < {Transparency,Composition,Analogical-Shape,Fit}.

The set of features on the left side of a "<" symbol is relaxed before the set on the right side. The order that the features inside the braces, "}...{", are relaxed is left unspecified (i.e., any order of relaxation is alright). Perceptual information about the domain also provides suggestions. Whenever a feature has feature values that are close, then one should be prepared to relax any of them to any of the others. In this example, since the colors are all very close – BLUE, TURQUOISE, and VIOLET – then Color may be a reasonable thing to relax. Hierarchical information about how closely related one feature value is to another can also be used to determine what to relax. The Shape values are a good example. A CYLINDRICAL shape is also a CONICAL shape, which is also a 3-D ROUND shape. Hence, it is very reasonable to match ROUNDED to CYLINDRICAL. All of these suggestions using perceptual information can be put together to form the order.

{Shape,Color} < {Subpart} < {Transparency,Composition,Analogical-Shape,Fit}.

The referent candidates *MAINTUBE*, *STAND*, and *AIR CHAMBER* can be examined and possibly ordered for relaxation using the above feature ordering. For this example, however, none of the possible feature orders is better for relaxing the candidates. Hence, no one candidate stands out as the most likely referent.

While no ordering of the candidates was possible, the order generated to relax the features in the speaker's description can be used to guide the relaxation of each candidate. The procedures mentioned at the end of the last section come into use here. Generate-Similar-Shape-Values can determine that HEMISPHERICAL and CYLINDRICAL shapes of the *AIR CHAMBER* are close to the 3-D ROUND shape. This

holds equally true for the shapes of the *MAINTUBE* and the *STAND*. Generate-Similar-Color-Values next tries relaxing the Color TURQUOISE. It determines the colors BLUE and GREEN as the best alternates. Here only two clear winners exist - the *AIR CHAMBER* and the *STAND*. Subpart, Transparency, Analogical-Shape, and Composition provide no further help (though, the fact that the *AIR CHAMBER* has both CLEAR and OPAQUE subparts might put it slightly lower than the *STAND* whose subparts are all CLEAR. This difference, however, is not significant.). This leaves trial and error attempts to try to complete the FIT action. The one (if any) that fits - and fits loosely - is selected as the referent. The protocols showed that people often do just that - reducing their set of choices down as best they can and then taking each of the remaining choices and trying out the requested action on them.

7.4 Conclusion

Natural language interactions in the real world invite contextually poor utterances. This paper sketches the ideas behind an on-going effort to develop a reference identification and plan recognition mechanism that can exhibit more "human" tolerance of such utterances. Our goal is to build a more robust system that can handle errorful utterances, and that is adaptable to existing systems.

The work attempts to provide a computational scheme for handling noun phrases (following the work on noun phrases by [15, 34, 24, 31]) that is robust enough to provide human-like performance. When people are asked to identify objects, they go about it in a certain way: find candidates, adjust as necessary, re-try, and, if necessary, give up and ask for help. We claim that relaxation is an integral part of this process and that the particular parameters of relaxation differ from task to task.

and person to person. Our work models the relaxation process and provides a computational model for experimenting with the different parameters.

ACKNOWLEDGEMENTS

I want to thank especially Candy Sidner for her insightful comments and suggestions during the course of this work. I'd also like to acknowledge the helpful comments of George Hadden, Diane Litman, Marc Vilain, Dave Waltz, Bonnie Webber and Bill Woods on this paper. Many thanks also to Phil Cohen, Scott Fertig and Kathy Starr for providing me with their water pump dialogues and for their invaluable observations on them.

REFERENCES

[1] Allen, James F.
A Plan-Based Approach to Speech Act Recognition.
PhD thesis, University of Toronto, 1979.

[2] Allen, James F., Alan M. Frisch, and Diane J. Litman.
ARGOT: The Rochester Dialogue System.
In *Proceedings of AAAI-82*, pages 66-70. Pittsburgh, Pa., August, 1982.

[3] Appelt, Douglas E.
Planning Natural Language Utterances to Satisfy Multiple Goals.
PhD thesis, Stanford University, 1981.

[4] Brachman, Ronald J.
A Structural Paradigm for Representing Knowledge.
PhD thesis, Harvard University, 1977
Also, Technical Report No. 3605, Bolt Beranek and Newman Inc.

[5] Brown, John Seely and Kurt VanLehn.
Repair Theory: A Generative Theory of Bugs in Procedural Skills.
Cognitive Science 4(4):379-426, 1980.

[6] Clark, H. H. and C. Marshall.
Definite reference and mutual knowledge.
In Joshi, Webber and Sags (editor), *Elements of Discourse Understanding*, pages 10-64. Cambridge University Press, 1981.

[7] Cohen, Philip R.
On Knowing What to Say: Planning Speech Acts.
PhD thesis, University of Toronto, 1978.

[8] Cohen, Philip R.
The need for Referent Identification as a Planned Action.
In *Proceedings of IJCAI-81*, pages 31-35. Vancouver, B.C., Canada, August, 1981.

[9] Cohen, Philip R., Scott Fertig and Kathy Starr.
Dependencies of Discourse Structure on the Modality of Communication: Telephone vs. Teletype.
In *Proceedings of ACL*, pages 28-35. Toronto, Ont., Canada, June, 1982.

[10] Cohen, Philip R.
Pragmatics, Speaker-Reference, and the Modality of Communication.
FLAIR Technical Report , Fairchild Laboratory for Artificial Intelligence Research, 1984.

[11] Gentner, Dedre.
The Structure of Analogical Models in Science.
Technical Report, Bolt Beranek and Newman Inc., July, 1980.

[12] Goodman, Bradley A.
Miscommunication in Task-Oriented Dialogues.
KRNL Group Working Paper, Bolt Beranek and Newman Inc., April 1982.

[13] Goodman, Bradley A
Repairing Miscommunication: Relaxation in Reference.
In *Proceedings of AAAI-83*, pages 134-138. Washington, D.C., August, 1983.

[14] Goodman, Bradley A
Communication and Miscommunication.
PhD thesis, University of Illinois, Urbana, 1984.

[15] Grosz, Barbara J
The Representation and Use of Focus in Dialogue Understanding.
PhD thesis, University of California, Berkeley, 1977.
Also, Technical Note 151, Stanford Research Institute.

[16] Grosz, Barbara J.
Focusing and descriptions in natural language dialogues.
In Joshi, Webber and Sags (editor), *Elements of Discourse Understanding*, pages 84-105 Cambridge University Press, 1981.

[17] Joshi, Aravind K.
Mutual Beliefs in Question-Answer Systems.
In N. Smith (editor), Mutual Beliefs, pages 181-197 Academic Press, 1982.

[18] Lipkis, Thomas.
A KL-ONE Classifier
In *Proceedings of the 1981 KL-One Workshop*, pages 128-145. June, 1982.
Report No. 4842, Bolt Beranek and Newman Inc. Also Consul Note # 5.
USC Information Sciences Institute, October 1981.

[19] Litman, Diane
Discourse and Problem Solving.
Report No. 5338, Bolt Beranek and Newman Inc., July, 1983.
Also, TR130, University of Rochester, Dept. of Computer Science.

[20] Mark, William
Realization
In *Proceedings of the 1981 KL-One Workshop*, pages 78-89. June, 1982.
Report No. 4842, Bolt Beranek and Newman Inc.

[21] McKeown, Kathleen R.
Recursion in Text and Its Use in Language Generation
In *Proceedings of AAAI-83*, pages 270-273. Washington, D.C., August, 1983.

[22] Nadathur, Gopalan and Aravind K. Joshi.
Mutual Beliefs in Conversational Systems: Their Role in Referring Expressions.
In *Proceedings of IJCAI-83*, pages 603-605. Karlsruhe, West Germany, August, 1983.

[23] Perrault, C. Raymond and Philip R. Cohen.
It's for your own good: a note on inaccurate reference.
In Joshi, Webber and Sags (editor), *Elements of Discourse Understanding*, pages 217-230.Cambridge University Press, 1981.

[24] Reichman, Rachel.
Conversational Coherency.
Cognitive Science 2(4):283-327, 1978.

[25] Reichman, Rachel
Plain Speaking: A Theory and Grammar of Spontaneous Discourse.
PhD thesis, Harvard University, 1981.
Also, Technical Report No. 4861, Bolt Beranek and Newman Inc.

[26] Ringle, Martin and Bertram Bruce.
Conversation Failure
In W. Lehnart and M. Ringle (editor), *Knowledge Representation and Natural Language Processing* Lawrence Erlbaum Associates, 1981

[27] Robinson, Ann E.
Determining Verb Phrase Referents in Dialogs.
American Journal of Computational Linguistics 7(1):1-16, 1981.

[28] Rubin, Andee
A Theoretical Taxonomy of the Differences between Oral and Written Language.
In Rand J. Spiro, Bertram C. Bruce, and William F. Brewer (editor), *Theoretical Issues in Reading Understanding*.Lawrence Erlbaum Associates, 1980.

[29] John R. Searle.
Speech Acts
Cambridge University Press, 1969.

[30] Sidner, C.L., and Israel, D.J.
Recognizing intended meaning and speaker's plans.
In *Proceedings of the International Joint Conference in Artificial Intelligence*, pages 203-208. The International Joint Conferences on Artifical Intelligence, The International Joint Conferences on Artifical Intelligence, Vancouver, B.C., August, 1981

[31] Sidner, Candace Lee
Towards a Computational Theory of Definite Anaphora Comprehension in English Discourse
PhD thesis, Massachusetts Institute of Technology, 1979.
Also, Report No. TR-537, MIT AI Lab.

[32] Sidner, C. L., M. Bates, R. J. Bobrow, R. J Brachman, P. R. Cohen, D. J. Israel, J. Schmolze, B. L. Webber, W A Woods.
Research in Knowledge Representation for Natural Language Understanding
Report No 4785. Bolt Beranek and Newman Inc., November, 1981

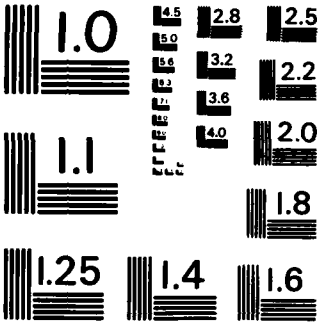
[33] Sidner, C L., Bates, M., Bobrow, R., Goodman, B., Haas, A., Ingria, R., Israel, D., McAllester, D., Moser, M., Schmolze, J., Vilain, M
Research in Knowledge Representation for Natural Language Understanding - Annual Report, 1 September 1982 - 31 August 1983
Technical Report 5421, BBN Laboratories, Cambridge, MA, 1983

[34] Webber, Bonnie Lynn.
A Formal Approach to Discourse Anaphora
PhD thesis, Harvard University, 1978.
Also, Technical Report No. 3761, Bolt Beranek and Newman Inc

[35] Winograd, Terry.
Procedures as a Representation for Data in a Computer Program for Understanding Natural Language
PhD thesis, Massachusetts Institute of Technology, 1971.
Also, Report No TR-84, Project MAC, MIT.

AD-A145 807 RESEARCH IN KNOWLEDGE REPRESENTATION FOR NATURAL LANGUAGE UNDERSTANDING(U) BOLT BERANEK AND NEWMAN INC CAMBRIDGE MA C SIDNER ET AL. SEP 84 BBN-5694 3/3

UNCLASSIFIED N00014-77-C-0378 F/G 5/7 NL



8. PUBLICATIONS

To conclude this report, we present a list of publications and presentations by members of the research group.

Papers Published

Israel, D. A Companion to the Naive Physics Manifesto. To appear as a chapter in *Formal Theories of the Common Sense World*, edited by Jerry Hobbs, to be published by Ablex in 1984 (in press)

Israel, D. Some Remarks on the Place of Logic in Knowledge Representation. Special issue of *IEEE Computer* on Knowledge Representation in Artificial Intelligence, October 1983, Vol. 16, No. 10, ps. 37-42. Also published as Technical Report No. 5388, Bolt Beranek and Newman Inc., Cambridge, MA, July 1983.

Sidner, C.L. Focusing and Discourse. In **Discourse Processes** 6, 1983, pp. 107-130.

Sidner, C.L. What the Speaker Means. The Recognition of Speaker's Plans in Discourse. In **International Journal of Computers and Mathematics**, Vol. 9, No. 1, 1983.

Sondheimer, N., Weischedel, R. and Bobrow, R. Semantic Interpretation using KL-1, to appear in **Proceedings of COLING, 1984**.

Presentations

Bates, M. Natural Language Interfaces to Database Systems. Computer Science Department, Duke University, October 27, 1983

Bates, M. Position paper, There Still is Gold in the Database Mine, for Panel on Natural Language and Databases, COLING84, Stanford University, July 1984.

Bates, M. Invited to attend Second International Workshop on Language Generation, Stanford, CA, July 8-10, 1984.

Bobrow, R. Natural Language Interfaces. What's Here, What's Coming, and Who Needs Them. Invited talk at ACM Northeast Regional Meeting, March 1984.

Bobrow, R Transportable Natural Language Database Interface. Invited talk at University of Delaware, May 1984.

Ingria, R Finiteness and "Copy Raising" in English and Modern Greek. Harvard University, December 15, 1983.

Israel, D. Some Remarks on the Place of Logic in Knowledge Representation. The Computer Science Colloquium at the University of Toronto, March 1983.

Israel, D Commentator on "Using Semantics in Non-Context-Free Parsing of Montague Grammar," by David S Warren and Joyce Friedman. Cognitive Science Center Seminar, MIT, October, 1983.

Israel, D. Situation Semantics, Model Structures and Set Theory Computer Science Colloquium, Boston University, November, 1983

Israel, D Kripke-Platek Set Theory Some Set Theory Some Recursion Theory. Seminar on the Foundations of Situation Semantics. The Center for the Study of Language and Information, Stanford University, April 1984

Israel, D. Laughably Weak Logics. A Shaky Leg Up On The Problem of Belief (?) Seminar. Department of Computer and Information Sciences, University of Pennsylvania, May 1984

Schmolze, J. From KL-ONE to KL-TWO, a New Hybrid Knowledge Representation System. Artificial Intelligence Center, Naval Research Laboratories, March 26, 1984

Forthcoming Papers

Goodman, B. Communication and Miscommunication. Ph.D. Thesis, Department of Computer Science, University of Illinois, Urbana (degree expected October 1984). BBN Report 5681, forthcoming.

Brachman, R. and Schmolze, J. An Overview of the KL-One Knowledge Representation System. In **Cognitive Science** to appear in 1985

Ingria, R. Complement Types in English. BBN Report No. 5684, forthcoming.

Official Distribution List

Contract N00014-77-C-0378

	Copies
Defense Documentation Center Cameron Station Alexandria, VA 22314	12
Office of Naval Research Information Systems Program Code 437 Arlington, VA 22217	2
Office of Naval Research Code 200 Arlington, VA 22217	1
Office of Naval Research Code 455 Arlington, VA 22217	1
Office of Naval Research Code 458 Arlington, VA 22217	1
Office of Naval Research Branch Office, Boston 495 Summer Street Boston, MA 02210	1
Office of Naval Research Branch Office, Chicago 536 South Clark Street Chicago, IL 60605	1
Office of Naval Research Branch Office, Pasadena 1030 East Green Street Pasadena, CA 91106	1
Naval Research Laboratory Technical Information Division Code 2627 Washington, D.C. 20380	6

cont'd.

Naval Ocean Systems Center 1
Advanced Software Technology Division
Code 5200
San Diego, CA 92152

Dr. A. L. Slafkosky 1
Scientific Advisor
Commandant of the Marine Corps
(Code RD-1)
Washington, D.C. 20380

Mr. E. H. Gleissner 1
Naval Ship Research & Development Ctr.
Computation & Mathematics Dept.
Bethesda, MD 20084

Capt. Grace M. Hopper, USNR 1
Naval Data Automation Command
Code 00H
Washington Navy Yard
Washington, D.C. 20374

Mr. Paul M. Robinson, Jr. 1
NAVDAC 33
Washington Navy Yard
Washington, D.C. 20374

Advanced Research Projects Agency 1
Information Processing Techniques
1400 Wilson Boulevard
Arlington, VA 22209

Capt. Richard L. Martin, USN 1
507 Breezy Point Crescent
Norfolk, VA 23511

Director, National Security Agency 1
Attn: R54, Mr. Page
Fort G.G. Meade, MD 20755

Director, National Security Agency 1
Attn: R54, Mr. Glick
Fort G.G. Meade, MD 20755

Major James R. Kreer 1
Chief, Information Sciences
Dept. of the Air Force
Air Force Office of Scientific Research
European Office of Aerospace
Research & Development
Box 14
FPO New York 09510

cont'd.

Mr. Fred M. Griffee 1
Technical Advisor C3 Division
Marine Corps Development
& Education Command
Quantico, VA 22134

END

FILMED

10-84

DTIC